



AT&T

305-646
Issue 1

AT&T 3B2 Computer
UNIX[®] System V Release 3

User's and System Administrator's
Reference Manual

**©1988 AT&T
All Rights Reserved
Printed in USA**

NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

DATAPHONE is a registered trademark of AT&T.
DEC is a registered trademark of Digital Equipment.
Diablo is a registered trademark of Xerox.
DOCUMENTER'S WORKBENCH is a trademark of AT&T.
HP is a registered trademark of Hewlett-Packard.
TEKTRONIX is a registered trademark of Tektronix.
TELETYPE is a registered trademark of AT&T.
TermiNet is a trademark of General Electric.
UNIX is a registered trademark of AT&T.
Versatec is a trademark of Versatec.

Introduction

This *User's Reference Manual* describes the commands that constitute the basic software running on the AT&T 3B2 Computer.

Several closely-related documents contain other valuable information:

- The *User's Guide* presents an overview of the UNIX system and tutorials on how to use text editors, automate repetitive jobs, and send information to others.
- The *Programmer's Guide* presents an overview of the UNIX system programming environment and tutorials on various programming tools.
- The *Programmer's Reference Manual* describes the commands, system calls, subroutines, libraries, file formats, and miscellaneous information used by programmers.
- The *System Administrator's Guide* provides procedures for and explanations of administrative tasks.
- The *System Administrator's Reference Manual* describes commands, file formats, and miscellaneous information used by system administrators.

Although the commands are each part of a specific Utilities Package listed below, they appear in alphabetical order in a single section of this document called "Commands." For a list of which commands are in each Utilities Package, see the "Index to Utilities" at the back of this manual.

1. AT&T Windowing Utilities
2. Basic Networking Utilities
3. Cartridge Tape Controller Utilities
4. Directory and File Management Utilities
5. Editing Utilities
6. Essential Utilities
7. Form and Menu Language Interpreter Utilities
8. Framed Access Command Environment Utilities
9. Inter-process Communications Utilities
10. Line Printer Spooling Utilities
11. Security Administration Utilities
12. Spell Utilities
13. Terminal Information Utilities
14. User Environment Utilities

Security Administration Utilities are expressly provided for U. S. customers.

Section (1): Commands

The entries in Section (1) describe programs intended to be invoked directly by the user or by command language procedures, as opposed to subroutines, which are called by the user's programs. Commands generally reside in the directory `/bin` (for binary programs). In addition, some commands reside in `/usr/bin`. These directories are searched automatically by the command interpreter called the *shell*. Also, UNIX systems often have a directory called `/usr/lbin`, containing local commands.

Throughout this manual, numbers following a command are intended for easy cross-reference. A command followed by a (1), (1C), or (1G) usually means that it is described in this manual. (Section (1) commands appropriate for use by programmers are located in the *Programmer's Reference Manual*.) A command with a (1M), (7), or (8) following it means that the command is in the corresponding section of the *System Administrator's Reference Manual*. A command with a (2) or (3) following it means that the command is in the corresponding section of the *Programmer's Reference Manual*. A command with a (4) or (5) following it means that the command is in the corresponding section of the *Programmer's Reference Manual* and the *System Administrator's Reference Manual*.

Each entry in the Commands section appears under a single name shown at the upper corners of its page(s). Entries are alphabetized, with the exception of the *intro*(1) entry, which is first. Some entries may describe several commands. In such cases, the entry appears only once, alphabetized under its "primary" name, the name that appears at the upper corners of the page. The "secondary" commands are listed directly below their associated primary command. To learn which manual page describes a secondary command, locate its name in the middle column of the "Permuted Index" and follow across that line to the name of the manual page listed in the right column.

All entries are presented using the following format (though some of these headings might not appear in every entry):

- **NAME** gives the primary name (and secondary name(s), as the case may be) and briefly states its purpose.
- **SYNOPSIS** summarizes the usage of the program being described. A few explanatory conventions are used, particularly in the **SYNOPSIS**:
 - **Boldface** strings are literals and are to be typed just as they appear.
 - *Italic* strings usually represent substitutable argument and command names found elsewhere in the manual.
 - Square brackets **[]** around an argument indicate that the argument is optional. When an argument is given as "name" or "file," it always refers to a *file* name.
 - Ellipses ... are used to show that the previous argument may be repeated.
 - A final convention is used by the commands themselves. An argument beginning with a minus (-), plus (+), or an equal sign (=) is often taken to be a flag argument, even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with -, +, or =.
- **DESCRIPTION** discusses how to use these commands.
- **EXAMPLE(S)** gives example(s) of usage, where appropriate.
- **FILES** contains the file names that are referenced by the program.
- **EXIT CODES** discusses values set when the command terminates. The value set is available in the shell environment variable "?" (see *sh(1)*).
- **NOTES** gives information that may be helpful under the particular circumstances described.
- **SEE ALSO** offers pointers to related information.
- **DIAGNOSTICS** discusses the error messages that may be produced. Messages that are intended to be self-explanatory are not listed.
- **WARNINGS** discusses the limits or boundaries of the respective commands.
- **BUGS** lists known faults in software that have not been rectified. Occasionally, a suggested short-term remedy is also described.

Preceding Section 1 are a "Table of Contents" (listing both primary and secondary command entries) and a "Permuted Index." Each line of the "Table of Contents" contains the name of a manual page (with secondary entries, if they exist) and an abstract of that page. Each line of the "Permuted Index" represents a permutation (or sorting) of a line from the "Table of Contents" into three columns. Each line is arranged so that a keyword or phrase begins the middle column. Use the "Permuted Index" by searching this middle column for a topic or command. When you have found the entry you want, the right column of that line lists the name of the manual page on which information corresponding to that keyword may be found. The left column contains the remainder of the permutation that began in the middle column.

How to Get Started

This discussion provides the basic information you need to get started on the UNIX system: how to log in and log out, how to communicate through your terminal, and how to run a program. (See the *User's Guide* for a more complete introduction to the system.)

Logging In

You must connect to the UNIX system from a full-duplex ASCII terminal. You must also have a valid login ID, which may be obtained (together with how to access your UNIX system) from the administrator of your system. Common terminal speeds are 30, 120, 240, 480, 960, 1920, and 3840 characters per second (300, 1200, 2400, 4800, 9600, 19200, and 38400 baud). Some UNIX systems have different ways of accessing each available terminal speed, while other systems offer several speeds through a common access method. In the latter case, there is one "preferred" speed; if you access it from a terminal set to a different speed, you will be greeted by a string of meaningless characters. Keep hitting the BREAK, INTERRUPT, or ATTENTION key until the `login:` prompt appears.

Most terminals have a speed switch that should be set to the appropriate speed and a half-/full-duplex switch that should be set to full-duplex. When a connection has been established, the system types `login:`. You respond by typing your login ID followed by the RETURN key. If you have a password, the system asks for it but will not print, or "echo," it on the terminal. After you have logged in, the RETURN, NEW-LINE, and LINE-FEED keys all have equivalent meanings.

Make sure you type your login name in lowercase letters. Typing uppercase letters causes the UNIX system to assume that your terminal can generate only uppercase letters, and it will treat all letters as uppercase for the remainder of your login session. The shell will print a \$ on your screen when you have logged in successfully.

When you log in, a message-of-the-day may greet you before you receive your prompt. For more information, consult *login(1)*, which discusses the login sequence in more detail, and *stty(1)*, which tells you how to describe your terminal to the system. *profile(4)* (in the *System Administrator's Reference Manual*) explains how to accomplish this last task automatically every time you log in.

Logging Out

There are two ways to log out:

- If you've dialed in, you can simply hang up the phone.
- You can log out by typing an end-of-file indication (ASCII EOT character, usually typed as CONTROL-D) to the shell. The shell will terminate, and the **login:** message will appear again.

How to Communicate Through Your Terminal

When you type to the UNIX system, your individual characters are being gathered and temporarily saved. Although they are echoed back to you, these characters will not be given to a program until you type a RETURN (or NEW-LINE) as described above in "Logging In."

UNIX system terminal input/output is full duplex. It has full read-ahead, which means that you can type at any time, even while a program is typing at you. Of course, if you type during output, your input characters will have output characters interspersed among them. In any case, whatever you type will be saved and interpreted in the correct sequence. There is a limit to the amount of read-ahead, but it is not likely to be exceeded.

The character @ cancels all the characters typed before it on a line, effectively deleting the line. (@ is called the "line kill" character.) The character # erases the last character typed. Successive uses of # will erase characters back to, but not beyond, the beginning of the line; @ and # can be typed as themselves by preceding them with \ (thus, to erase a \, you need two #s). These default erase and line kill characters can be changed; see *stty(1)*.

CONTROL-S (also known as the ASCII DC3 character) is typed by pressing the CONTROL key and the alphabetic **s** simultaneously; it is used to stop output temporarily. It is useful with CRT terminals to prevent output from disappearing before it can be read. Output is resumed when a CONTROL-Q (also known as DC1) is typed. Thus, if you had typed **cat yourfile** and the contents of **yourfile** were passing by on the screen more rapidly than you could read it, you would type CONTROL-S to freeze the output. Typing CONTROL-Q would allow the output to resume. The CONTROL-S and CONTROL-Q characters are not passed to any other program when used in this manner.

The ASCII DEL (also called "rubout") character is not passed to programs but instead generates an interrupt signal, just like the BREAK, INTERRUPT, or ATTENTION signal. This signal generally causes whatever program you are running to terminate. It is typically used to stop a long printout that you do not want. Programs, however, can arrange either to ignore this signal altogether or to be notified and take a specific action when it happens (instead of being terminated). The editor *ed*(1), for example, catches interrupts and stops what *it* is doing, instead of terminating, so an interrupt can be used to halt an editor printout without losing the file being edited.

Besides adapting to the speed of the terminal, the UNIX system tries to be intelligent as to whether you have a terminal with the NEW-LINE function, or whether it must be simulated with a CARRIAGE-RETURN and LINE-FEED pair. In the latter case, all *input* CARRIAGE-RETURN characters are changed to LINE-FEED characters (the standard line delimiter), and a CARRIAGE-RETURN and LINE-FEED pair is echoed to the terminal. If you get into the wrong mode, the *stty*(1) command will rescue you.

Tab characters are used freely in UNIX system source programs. If your terminal does not have the tab function, you can arrange to have tab characters changed into spaces during output, and echoed as spaces during input. Again, the *stty*(1) command will set or reset this mode. The system assumes that tabs are set every eight character positions. The *tabs*(1) command will set tab stops on your terminal, if that is possible.

How to Run a Program

When you have successfully logged into the UNIX system, a program called the shell is communicating with your terminal. The shell reads each line you type, splits the line into a command name and its arguments, and executes the command. A command is simply an executable program.

Normally, the shell looks first in your current directory (see "The Current Directory" below) for the named program and, if none is there, then in system directories, such as `/bin` and `/usr/bin`. There is nothing special about system-provided commands except that they are kept in directories where the shell can find them. You can also keep commands in your own directories and instruct the shell to find them there. See the manual entry for `sh(1)`, under the sub-heading "Parameter Substitution," for the discussion of the `PATH` shell environmental variable.

The command name is the first word on an input line to the shell; the command and its arguments are separated from one another by space or tab characters.

When a program terminates, the shell will ordinarily regain control and give you back your prompt to indicate that it is ready for another command. The shell has many other capabilities, which are described in detail in `sh(1)`.

The Current Directory

The UNIX system has a file system arranged in a hierarchy of directories. When you received your login ID, the system administrator also created a directory for you (ordinarily with the same name as your login ID, and known as your login or home directory). When you log in, that directory becomes your current or working directory, and any file name you type is, by default, assumed to be in that directory. Because you are the owner of this directory, you have full permissions to read, write, alter, or remove its contents. Permissions to enter or modify other directories and files will have been granted or denied to you by their respective owners or by the system administrator. To change the current directory, use `cd(1)`.

Pathnames

To refer to files or directories not in the current directory, you must use a pathname. Full pathnames begin with `/`, which is the name of the root directory of the whole file system. After the slash comes the name of each directory containing the next subdirectory (followed by a `/`), until finally the file or directory name is reached (for example, `/usr/ae/filex` refers to file `filex` in directory `ae`, while `ae` is itself a subdirectory of `usr`, and `usr` is a subdirectory of the root directory). Use `pwd(1)` to print the full pathname of the directory you are working in. See `intro(2)` in the *Programmer's Reference Manual* for a formal definition of *pathname*.

If your current directory contains subdirectories, the pathnames of their respective files begin with the name of the corresponding subdirectory (without a prefixed `/`). A pathname may be used anywhere a file name is required.

Important commands that affect files are `cp(1)`, `mv` (see `cp(1)`), and `rm(1)`, which respectively copy, move (that is, rename), and remove files. To find out the status of files or directories, use `ls(1)`. Use `mkdir(1)` for making directories and `rmdir` (see `rm(1)`) for removing them.

Text Entry and Display

Almost all text is entered through an editor. Common examples of UNIX system editors are `ed(1)` and `vi(1)`. The commands most often used to print text on a terminal are `cat(1)`, `pr(1)`, and `pg(1)`. The `cat(1)` command displays the contents of ASCII text files on the terminal, with no processing at all. The `pr(1)` command paginates the text, supplies headings, and has a facility for multi-column output. The `pg(1)` command displays text in successive portions no larger than your terminal screen.

Writing a Program

Once you have entered the text of your program into a file with an editor, you are ready to give the file to the appropriate language processor. The processor will accept only files observing the correct naming conventions: all C programs must end with the suffix `.c`, and Fortran programs must end with `.f`. The output of the language processor will be left in a file named `a.out` in the current directory, unless you have invoked an option to save it in another file. (Use `mv(1)` to rename `a.out`.) If the program is written in assembly language, you will probably need to load library subroutines with it (see `ld(1)` in the *Programmer's Reference Manual*).

When you have completed this process without provoking any diagnostics, you may run the program by giving its name to the shell in response to the `$` prompt. Your programs can receive arguments from the command line just as system programs do; see `exec(2)` in the *Programmer's Reference Manual*. For more information on writing and running programs, see the *Programmer's Guide*.

Communicating with Others

Certain commands provide inter-user communication. Even if you do not plan to use them, it's helpful to learn something about them because someone else may try to contact you. *mail(1)* or *mailx(1)* will leave a message whose presence will be announced to another user when he or she next logs in and at periodic intervals during the session. To communicate with another user currently logged in, use *write(1)*. The corresponding entries in this manual also suggest how to respond to these two commands if you are their target.

See the tutorials in Chapter 8 of the *User's Guide* for more information on communicating with others.



Replace this Page
with the
TABLE OF CONTENTS

Tab Separator



Table of Contents

1. Commands

intro(1)	introduction to commands and application programs
ar(1)	archive and library maintainer for portable archives
assist(1)	assistance using UNIX system commands
astgen(1)	program for generating/modifying ASSIST menus or command forms
at, batch(1)	execute commands at a later time
awk(1)	pattern scanning and processing language
banner(1)	make posters
basename, dirname(1)	deliver portions of path names
bc(1)	arbitrary-precision arithmetic language
bdiff(1)	big diff
bfs(1)	big file scanner
cal(1)	print calendar
calendar(1)	reminder service
cat(1)	concatenate and print files
cd(1)	change working directory
chmod(1)	change mode
chown, chgrp(1)	change owner or group
cmp(1)	compare two files
col(1)	filter reverse line-feeds
comm(1)	select or reject lines common to two sorted files
cp, ln, mv(1)	copy, link or move files
cpio(1)	copy file archives in and out
crontab(1)	user crontab file
crypt(1)	encode/decode
csplit(1)	context split
ct(1C)	spawn getty to a remote terminal
cu(1C)	call another UNIX system
cut(1)	cut out selected fields of each line of a file
date(1)	print and set the date
dc(1)	desk calculator
dd(1M)	convert and copy a file
deroff(1)	remove nroff/troff, tbl, and eqn constructs
df(1M)	report number of free disk blocks and i-nodes
diff(1)	differential file comparator
diff3(1)	3-way differential file comparison
dircmp(1)	directory comparison
dconfig(1)	display data storage device configuration

Table of Contents

du(1M)	summarize disk usage
echo(1)	echo arguments
ed, red(1)	text editor
edit(1)	text editor (variant of ex for casual users)
egrep(1)	search a file for a pattern using full regular expressions
enable, disable(1)	enable/disable LP printers
env(1)	set environment for command execution
ex(1)	text editor
expr(1)	evaluate arguments as an expression
factor(1)	obtain the prime factors of a number
fgrep(1)	search a file for a character string
file(1)	determine file type
find(1)	find files
getopt(1)	parse command options
getopts, getoptcv(1)	parse command options
graph(1G)	draw a graph
grep(1)	search a file for a pattern
id(1M)	print user and group IDs and names
ipcrm(1)	remove a message queue, semaphore set or shared memory id
ipcs(1)	report inter-process communication facilities status
ismpx(1)	return windowing terminal state
join(1)	relational database operator
jterm(1)	reset layer of windowing terminal
jwin(1)	print size of layer
kill(1)	terminate a process
layers(1)	layer multiplexor for windowing terminals
line(1)	read one line
login(1)	sign on
logname(1)	get login name
lp, cancel(1)	send/cancel requests to an LP print service
lpstat(1)	print information about the status of the LP print service
ls(1)	list contents of directory
machid: pdp11, u3b, u3b2, u3b5, vax(1)	get processor type truth value
mail, rmail(1)	send mail to users or read mail
mailx(1)	interactive message processing system
makekey(1)	generate encryption key
mesg(1)	permit or deny messages
mkdir(1)	make directories
nawk(1)	pattern scanning and processing language
newform(1)	change the format of a text file
newgrp(1M)	log in to a new group

news(1) print news items

nice(1) run a command at low priority

nl(1) line numbering filter

nohup(1) run a command immune to hangsups and quits

od(1) octal dump

pack, pcat, unpack(1) compress and expand files

passwd(1) change login password and password attributes

paste(1) merge same lines of several files or subsequent lines of one file

pg(1) file perusal filter for CRTs

pr(1) print files

ps(1) report process status

pwd(1) working directory name

relogin(1M) rename login entry to show current layer

rm, rmdir(1) remove files or directories

sag(1G) system activity graph

sar(1) system activity reporter

sdiff(1) side-by-side difference program

sed(1) stream editor

setup(1) initialize system for first user

sh, rsh(1) shell, the standard/restricted command programming language

shl(1) shell layer manager

sleep(1) suspend execution for an interval

sort(1) sort and/or merge files

spell, hashmake, spellin, hashcheck(1) find spelling errors

split(1) split a file into pieces

stty(1) set the options for a terminal

su(1M) become super-user or another user

sum(1) print checksum and block count of a file

sync(1M) update the super block

sysadm(1) menu interface to do system administration

tabs(1) set tabs on a terminal

tail(1) deliver the last part of a file

tar(1) tape file archiver

tee(1) pipe fitting

test(1) condition evaluation command

time(1) time a command

timex(1) time a command; report process data and system activity

touch(1) update access and modification times of a file

tplot(1G) graphics filters

tput(1) initialize a terminal or query terminfo database

tr(1) translate characters

Table of Contents

true, false(1)	provide truth values
tty(1)	get the name of the terminal
umask(1)	set file-creation mode mask
uname(1)	print name of current UNIX system
uniq(1)	report repeated lines in a file
units(1)	conversion program
uucp, uulog, uuname(1C)	UNIX-to-UNIX system copy
uustat(1C)	uucp status inquiry and job control
uuto, uupick(1C)	public UNIX-to-UNIX system file copy
uux(1C)	UNIX-to-UNIX system command execution
vi(1)	screen-oriented (visual) display editor based on ex
wait(1)	await completion of process
wall(1)	write to all users
wc(1)	word count
who(1)	who is on the system
write(1)	write to another user
xargs(1)	construct argument list(s) and execute command

Permuted Index

diff3(1) 3-way differential file comparison diff3(1)
file touch(1) update access and modification times of a touch(1)
sag(1G) system activity graph sag(1G)
sar(1) system activity reporter sar(1)
report process data and system activity timex(1) time a command; timex(1)
menu interface to do system administration sysadm(1) sysadm(1)
sort(1) sort and/or merge files sort(1)
introduction to commands and application programs intro(1) intro(1)
maintainer for portable archives ar(1) archive and library ar(1)
language bc(1) arbitrary-precision arithmetic bc(1)
portable archives ar(1) archive and library maintainer for ar(1)
tar(1) tape file archiver tar(1)
and library maintainer for portable archives ar(1) archive ar(1)
cpio(1) copy file archives in and out cpio(1)
command xargs(1) construct argument list(s) and execute xargs(1)
expr(1) evaluate arguments as an expression expr(1)
echo(1) echo arguments echo(1)
bc(1) arbitrary-precision arithmetic language bc(1)
/program for generating/modifying ASSIST menus or command forms astgen(1)
system commands assist(1) assistance using UNIX assist(1)
commands assist(1) assistance using UNIX system assist(1)
generating/modifying ASSIST menus/ astgen(1) program for astgen(1)
a later time at(1) batch(1) execute commands at at(1)
change login password and password attributes passwd(1) passwd(1)
wait(1) await completion of process wait(1)
processing language awk(1) pattern scanning and awk(1)
banner(1) make posters banner(1)
(visual) display editor based on ex vi(1) screen-oriented vi(1)
portions of path names basename(1) dirname(1) deliver basename(1)
later time at(1) batch(1) execute commands at a at(1)
arithmetic language bc(1) arbitrary-precision bc(1)
su(1M) become super-user or another user su(1M)
bfs(1) big file scanner bfs(1)
bdiff(1) big diff bdiff(1)
bfs(1) big file scanner bfs(1)
sum(1) print checksum and block count of a file sum(1)
sync(1M) update the super block sync(1M)
df(1M) report number of free disk blocks and i-nodes df(1M)
cal(1) print calendar cal(1)
calculator dc(1)
calendar cal(1)
calendar(1) reminder service calendar(1)
cu(1C) call another UNIX system cu(1C)
an LP print service lp(1) cancel(1) send/cancel requests to lp(1)
text editor (variant of ex for casual users) edit(1) edit(1)
cat(1) concatenate and print files cat(1)

Permuted Index

attributes passwd(1)
 chmod(1)
 chown(1) chgrp(1)
 newform(1)
 cd(1)
fgrep(1) search a file for a
 tr(1) translate
 sum(1) print
 chown(1)
 group
common to two sorted files
 nice(1) run a
 env(1) set environment for
uux(1C) UNIX-to-UNIX system
 ASSIST menus or
 nohup(1) run a
 getopt(1) parse
getopts(1) getoptcv(1) parse
/shell, the standard/restricted
system activity timex(1) time a
 test(1) condition evaluation
 time(1) time a
argument list(s) and execute
 intro(1) introduction to
assistance using UNIX system
 at(1) batch(1) execute
comm(1) select or reject lines
 ipcs(1) report inter-process
 diff(1) differential file
 cmp(1)
diff3(1) 3-way differential file
 dircmp(1) directory
 wait(1) await
 pack(1) pcat(1) unpack(1)
 cat(1)
 test(1)
display data storage device
execute command xargs(1)
remove nroff/troff, tbl, and eqn
 ls(1) list
 csplit(1)
uucp status inquiry and job
 units(1)
 dd(1M)
 dd(1M) convert and
 cd(1) change working directory cd(1)
 change login password and password passwd(1)
 change mode chmod(1)
 change owner or group chown(1)
 change the format of a text file newform(1)
 change working directory cd(1)
 character string fgrep(1)
 characters tr(1)
 checksum and block count of a file sum(1)
 chgrp(1) change owner or group chown(1)
 chmod(1) change mode chmod(1)
 chown(1) chgrp(1) change owner or chown(1)
 cmp(1) compare two files cmp(1)
 col(1) filter reverse line-feeds col(1)
 comm(1) select or reject lines comm(1)
 command at low priority nice(1)
 command execution env(1)
 command execution uux(1C)
 command forms /generating/modifying astgen(1)
 command immune to hangups and quits nohup(1)
 command options getopt(1)
 command options getopts(1)
 command programming language sh(1)
 command; report process data and timex(1)
 command test(1)
 command time(1)
 command xargs(1) construct xargs(1)
 commands and application programs intro(1)
 commands assist(1) assist(1)
 commands at a later time at(1)
 common to two sorted files comm(1)
 communication facilities status ipcs(1)
 comparator diff(1)
 compare two files cmp(1)
 comparison diff3(1)
 comparison dircmp(1)
 completion of process wait(1)
 compress and expand files pack(1)
 concatenate and print files cat(1)
 condition evaluation command test(1)
 configuration dsconfig(1) dsconfig(1)
 construct argument list(s) and xargs(1)
 constructs deroff(1) deroff(1)
 contents of directory ls(1)
 context split csplit(1)
 control uustat(1C) uustat(1C)
 conversion program units(1)
 convert and copy a file dd(1M)
 copy a file dd(1M)

cpio(1) copy file archives in and out cpio(1)
 cp(1) ln(1) mv(1) copy, link or move files cp(1)
 uuname(1C) UNIX-to-UNIX system copy uucp(1C) uulog(1C) uucp(1C)
 public UNIX-to-UNIX system file copy uuto(1C) uupick(1C) uuto(1C)
 sum(1) print checksum and block count of a file sum(1)
 wc(1) word count wc(1)
 move files cp(1) ln(1) mv(1) copy, link or cp(1)
 out cpio(1) copy file archives in and cpio(1)
 crontab(1) user crontab file crontab(1)
 crontab(1) user crontab file crontab(1)
 pg(1) file perusal filter for CRTs pg(1)
 crypt(1) encode/decode crypt(1)
 csplit(1) context split csplit(1)
 terminal ct(1C) spawn getty to a remote ct(1C)
 cu(1C) call another UNIX system cu(1C)
 rename login entry to show current layer relogin(1M) relogin(1M)
 uname(1) print name of current UNIX system uname(1)
 line of a file cut(1) cut out selected fields of each cut(1)
 each line of a file cut(1) cut out selected fields of cut(1)
 time a command; report process data and system activity timex(1) timex(1)
 dsconfig(1) display data storage device configuration dsconfig(1)
 join(1) relational database operator join(1)
 a terminal or query terminfo database tput(1) initialize tput(1)
 date(1) print and set the date date(1)
 date(1) print and set the date date(1)
 dc(1) desk calculator dc(1)
 dd(1M) convert and copy a file dd(1M)
 basename(1) dirname(1) deliver portions of path names basename(1)
 tail(1) deliver the last part of a file tail(1)
 msg(1) permit or deny messages msg(1)
 and eqn constructs deroff(1) remove nroff/troff, tbl, deroff(1)
 dc(1) desk calculator dc(1)
 file(1) determine file type file(1)
 dsconfig(1) display data storage device configuration dsconfig(1)
 blocks and i-nodes df(1M) report number of free disk df(1M)
 bdiff(1) big diff bdiff(1)
 comparator diff(1) differential file diff(1)
 comparison diff3(1) 3-way differential file diff3(1)
 sdiff(1) side-by-side difference program sdiff(1)
 diff(1) differential file comparator diff(1)
 diff3(1) 3-way differential file comparison diff3(1)
 mkdir(1) make dircmp(1) directory comparison dircmp(1)
 directories mkdir(1)
 rm(1) rmdir(1) remove files or directories rm(1)
 cd(1) change working directory cd(1)
 dircmp(1) directory comparison dircmp(1)
 ls(1) list contents of directory ls(1)
 pwd(1) working directory name pwd(1)
 names basename(1) dirname(1) deliver portions of path basename(1)

Permuted Index

printers enable(1) disable(1) enable/disable LP enable(1)
df(1M) report number of free disk blocks and i-nodes df(1M)
du(1M) summarize disk usage du(1M)
configuration dsconfig(1) display data storage device dsconfig(1)
vi(1) screen-oriented (visual) display editor based on ex vi(1)
graph(1G) draw a graph graph(1G)
device configuration dsconfig(1) display data storage dsconfig(1)
du(1M) summarize disk usage du(1M)
od(1) octal dump od(1)
echo(1) echo arguments echo(1)
echo(1) echo arguments echo(1)
ed(1) red(1) text editor ed(1)
for casual users) edit(1) text editor (variant of ex edit(1)
screen-oriented (visual) display editor based on ex vi(1) vi(1)
ed(1) red(1) text editor ed(1)
ex(1) text editor ex(1)
sed(1) stream editor sed(1)
users) edit(1) text editor (variant of ex for casual edit(1)
pattern using full regular/ egrep(1) search a file for a egrep(1)
LP printers enable(1) disable(1) enable/disable enable(1)
enable(1) disable(1) enable/disable LP printers enable(1)
crypt(1) encode/decode crypt(1)
makekey(1) generate encryption key makekey(1)
relogin(1M) rename login entry to show current layer relogin(1M)
execution env(1) set environment for command env(1)
env(1) set environment for command execution env(1)
remove nroff/troff, tbl, and eqn constructs deroff(1) deroff(1)
hashcheck(1) find spelling errors /hashmake(1) spellin(1) spell(1)
expr(1) evaluate arguments as an expression expr(1)
test(1) condition evaluation command test(1)
edit(1) text editor (variant of ex for casual users) edit(1)
(visual) display editor based on ex vi(1) screen-oriented vi(1)
ex(1) text editor ex(1)
construct argument list(s) and execute command xargs(1) xargs(1)
at(1) batch(1) execute commands at a later time at(1)
env(1) set environment for command execution env(1)
sleep(1) suspend execution for an interval sleep(1)
uux(1C) UNIX-to-UNIX system command execution uux(1C)
pcat(1) unpack(1) compress and expand files pack(1) pack(1)
expression expr(1) evaluate arguments as an expr(1)
expr(1) evaluate arguments as an expression expr(1)
for a pattern using full regular expressions egrep(1) search a file egrep(1)
report inter-process communication facilities status ipcs(1) ipcs(1)
of a number factor(1) obtain the prime factors factor(1)
factor(1) obtain the prime factors of a number factor(1)
true(1) false(1) provide truth values true(1)
character string fgrep(1) search a file for a fgrep(1)
cut(1) cut out selected fields of each line of a file cut(1)
tar(1) tape file archiver tar(1)

cpio(1) copy	file archives in and out	cpio(1)
diff(1) differential	file comparator	diff(1)
diff3(1) 3-way differential	file comparison	diff3(1)
public UNIX-to-UNIX system	file copy uuto(1C) uupick(1C)	uuto(1C)
crontab(1) user crontab	file	crontab(1)
selected fields of each line of a	file cut(1) cut out	cut(1)
dd(1M) convert and copy a	file	dd(1M)
fgrep(1) search a	file for a character string	fgrep(1)
grep(1) search a	file for a pattern	grep(1)
regular/ egrep(1) search a	file for a pattern using full	egrep(1)
split(1) split a	file into pieces	split(1)
change the format of a text	file newform(1)	newform(1)
files or subsequent lines of one	file /merge same lines of several	paste(1)
pg(1)	file perusal filter for CRTs	pg(1)
bfs(1) big	file scanner	bfs(1)
print checksum and block count of a	file sum(1)	sum(1)
tail(1) deliver the last part of a	file	tail(1)
access and modification times of a	file touch(1) update	touch(1)
file(1) determine	file type	file(1)
uniq(1) report repeated lines in a	file	uniq(1)
umask(1) set	file(1) determine file type	file(1)
cat(1) concatenate and print	file-creation mode mask	umask(1)
cmp(1) compare two	files	cat(1)
reject lines common to two sorted	files	cmp(1)
ln(1) mv(1) copy, link or move	files comm(1) select or	comm(1)
find(1) find	files cp(1)	cp(1)
rm(1) rmdir(1) remove	files	find(1)
file /merge same lines of several	files or directories	rm(1)
unpack(1) compress and expand	files or subsequent lines of one	paste(1)
pr(1) print	files pack(1) pcat(1)	pack(1)
sort(1) sort and/or merge	files	pr(1)
pg(1) file perusal	files	sort(1)
nl(1) line numbering	filter for CRTs	pg(1)
col(1)	filter	nl(1)
tplot(1G) graphics	filter reverse line-feeds	col(1)
find(1)	filters	tplot(1G)
hashmake(1) spellin(1) hashcheck(1)	find files	find(1)
tee(1) pipe	find spelling errors spell(1)	spell(1)
newform(1) change the	find(1) find files	find(1)
ASSIST menus or command	fitting	tee(1)
df(1M) report number of	format of a text file	newform(1)
search a file for a pattern using	forms /for generating/modifying	astgen(1)
makekey(1)	free disk blocks and i-nodes	df(1M)
or command/ astgen(1) program for	full regular expressions egrep(1)	egrep(1)
getopts(1)	generate encryption key	makekey(1)
command options	generating/modifying ASSIST menus	astgen(1)
	getopt(1) parse command options	getopt(1)
	getoptcv(1) parse command options	getopts(1)
	getopts(1) getoptcv(1) parse	getopts(1)

Permuted Index

ct(1C) spawn
graph(1G) draw a
sag(1G) system activity
tplot(1G)
chown(1) chgrp(1) change owner or
id(1M) print user and
newgrp(1M) log in to a new
nohup(1) run a command immune to
spell(1) hashmake(1) spellin(1)
find spelling errors spell(1)
semaphore set or shared memory
names
id(1M) print user and group
nohup(1) run a command
LP print service lpstat(1) print
terminfo database tput(1)
setup(1)
number of free disk blocks and
uustat(1C) uucp status
system mailx(1)
administration sysadm(1) menu
facilities status ipcs(1) report
sleep(1) suspend execution for an
and application programs
application programs intro(1)
semaphore set or shared memory id
communication facilities status
state
news(1) print news
uustat(1C) uucp status inquiry and
operator
terminal
makekey(1) generate encryption
pattern scanning and processing
arbitrary-precision arithmetic
pattern scanning and processing
command programming
batch(1) execute commands at a
jwin(1) print size of
shl(1) shell
terminals layers(1)
jterm(1) reset
rename login entry to show current
windowing terminals
archives ar(1) archive and
getty to a remote terminal ct(1C)
graph graph(1G)
graph sag(1G)
graph(1G) draw a graph graph(1G)
graphics filters tplot(1G)
grep(1) search a file for a pattern grep(1)
group chown(1)
group IDs and names id(1M)
group newgrp(1M)
hangups and quits nohup(1)
hashcheck(1) find spelling errors spell(1)
hashmake(1) spellin(1) hashcheck(1) spell(1)
id /remove a message queue, ipcrm(1)
id(1M) print user and group IDs and id(1M)
IDs and names id(1M)
immune to hangups and quits nohup(1)
information about the status of the lpstat(1)
initialize a terminal or query tput(1)
initialize system for first user setup(1)
i-nodes df(1M) report df(1M)
inquiry and job control uustat(1C)
interactive message processing mailx(1)
interface to do system sysadm(1)
inter-process communication ipcs(1)
interval sleep(1)
intro(1) introduction to commands intro(1)
introduction to commands and intro(1)
ipcrm(1) remove a message queue, ipcrm(1)
ipcs(1) report inter-process ipcs(1)
ismpx(1) return windowing terminal ismpx(1)
items news(1)
job control uustat(1C)
join(1) relational database join(1)
jterm(1) reset layer of windowing jterm(1)
jwin(1) print size of layer jwin(1)
key makekey(1)
kill(1) terminate a process kill(1)
language awk(1) awk(1)
language bc(1) bc(1)
language nawk(1) nawk(1)
language /the standard/restricted sh(1)
later time at(1) at(1)
layer jwin(1)
layer manager shl(1)
layer multiplexor for windowing layers(1)
layer of windowing terminal jterm(1)
layer relogin(1M) relogin(1M)
layers(1) layer multiplexor for layers(1)
library maintainer for portable ar(1)

line(1) read one	line	line(1)
nl(1)	line numbering filter	nl(1)
cut out selected fields of each	line of a file cut(1)	cut(1)
col(1) filter reverse	line(1) read one line	line(1)
comm(1) select or reject	line-feeds	col(1)
uniq(1) report repeated	lines common to two sorted files	comm(1)
of several files or subsequent	lines in a file	uniq(1)
subsequent/ paste(1) merge same	lines of one file /merge same lines	paste(1)
cp(1) ln(1) mv(1) copy,	lines of several files or	paste(1)
ls(1)	link or move files	cp(1)
xargs(1) construct argument	list contents of directory	ls(1)
files cp(1)	list(s) and execute command	xargs(1)
newgrp(1M)	ln(1) mv(1) copy, link or move	cp(1)
relogin(1M) rename	log in to a new group	newgrp(1M)
logname(1) get	login entry to show current layer	relogin(1M)
attributes passwd(1) change	login name	logname(1)
	login password and password	passwd(1)
	login(1) sign on	login(1)
	logname(1) get login name	logname(1)
	low priority	nice(1)
nice(1) run a command at	LP print service lp(1) cancel(1)	lp(1)
send/cancel requests to an	LP print service lpstat(1) print	lpstat(1)
information about the status of the	LP printers	enable(1)
enable(1) disable(1) enable/disable	lp(1) cancel(1) send/cancel	lp(1)
requests to an LP print service	lpstat(1) print information about	lpstat(1)
the status of the LP print service	ls(1) list contents of directory	ls(1)
	machid(1) pdp11(1) u3b(1) u3b2(1)	machid(1)
u3b5(1) vax(1) get processor type/	mail mail(1)	mail(1)
rmail(1) send mail to users or read	mail to users or read mail	mail(1)
mail(1) rmail(1) send	mail(1) rmail(1) send mail to users	mail(1)
or read mail	mailx(1) interactive message	mailx(1)
processing system	maintainer for portable archives	ar(1)
ar(1) archive and library	makekey(1) generate encryption key	makekey(1)
	manager	shl(1)
shl(1) shell layer	mask	umask(1)
umask(1) set file-creation mode	memory id /remove a message	ipcrm(1)
queue, semaphore set or shared	menu interface to do system	sysadm(1)
administration sysadm(1)	menus or command forms /program	astgen(1)
for generating/modifying ASSIST	merge files	sort(1)
sort(1) sort and/or	merge same lines of several files	paste(1)
or subsequent lines of/ paste(1)	mesg(1) permit or deny messages	mesg(1)
	message processing system	mailx(1)
mailx(1) interactive	message queue, semaphore set or	ipcrm(1)
shared memory id ipcrm(1) remove a	messages	mesg(1)
mesg(1) permit or deny	mkdir(1) make directories	mkdir(1)
	mode	chmod(1)
chmod(1) change	mode mask	umask(1)
umask(1) set file-creation	modification times of a file	touch(1)
touch(1) update access and	move files	cp(1)
cp(1) ln(1) mv(1) copy, link or		

Permuted Index

layers(1) layer	multiplexor for windowing terminals	layers(1)
cp(1) ln(1)	mv(1) copy, link or move files	cp(1)
logname(1) get login	name	logname(1)
uname(1) print	name of current UNIX system	uname(1)
tty(1) get the	name of the terminal	tty(1)
pwd(1) working directory	name	pwd(1)
dirname(1) deliver portions of path	names basename(1)	basename(1)
id(1M) print user and group IDs and	names	id(1M)
processing language	nawk(1) pattern scanning and	nawk(1)
text file	newform(1) change the format of a	newform(1)
	newgrp(1M) log in to a new group	newgrp(1M)
news(1) print	news items	news(1)
	news(1) print news items	news(1)
priority	nice(1) run a command at low	nice(1)
	nl(1) line numbering filter	nl(1)
hangups and quits	nohup(1) run a command immune to	nohup(1)
constructs deroff(1) remove	nroff/troff, tbl, and eqn	deroff(1)
obtain the prime factors of a	number factor(1)	factor(1)
i-nodes df(1M) report	number of free disk blocks and	df(1M)
nl(1) line	numbering filter	nl(1)
number factor(1)	obtain the prime factors of a	factor(1)
od(1)	octal dump	od(1)
	od(1) octal dump	od(1)
join(1) relational database	operator	join(1)
stty(1) set the	options for a terminal	stty(1)
getopt(1) parse command	options	getopt(1)
getoptcv(1) parse command	options getopt(1)	getopts(1)
chown(1) chgrp(1) change	owner or group	chown(1)
and expand files	pack(1) pcat(1) unpack(1) compress	pack(1)
getopt(1)	parse command options	getopt(1)
getopts(1) getoptcv(1)	parse command options	getopts(1)
tail(1) deliver the last	part of a file	tail(1)
password attributes	passwd(1) change login password and	passwd(1)
passwd(1) change login	password and password attributes	passwd(1)
passwd(1) change login password and	password attributes	passwd(1)
several files or subsequent lines/	paste(1) merge same lines of	paste(1)
dirname(1) deliver portions of	path names basename(1)	basename(1)
grep(1) search a file for a	pattern	grep(1)
language awk(1)	pattern scanning and processing	awk(1)
language nawk(1)	pattern scanning and processing	nawk(1)
egrep(1) search a file for a	pattern using full regular/	egrep(1)
expand files pack(1)	pcat(1) unpack(1) compress and	pack(1)
vax(1) get processor/ machid(1)	pdp11(1) u3b(1) u3b2(1) u3b5(1)	machid(1)
mesg(1)	permit or deny messages	mesg(1)
pg(1) file	perusal filter for CRTs	pg(1)
	pg(1) file perusal filter for CRTs	pg(1)
split(1) split a file into	pieces	split(1)
tee(1)	pipe fitting	tee(1)
archive and library maintainer for	portable archives ar(1)	ar(1)

basename(1)	dirname(1)	deliver	portions of path names	basename(1)
	banner(1)	make	posters	banner(1)
	factor(1)	obtain the	pr(1) print files	pr(1)
		date(1)	prime factors of a number	factor(1)
		cal(1)	print and set the date	date(1)
		file sum(1)	print calendar	cal(1)
	cat(1)	concatenate and	print checksum and block count of a	sum(1)
		pr(1)	print files	cat(1)
		pr(1)	print files	pr(1)
of the LP print service	lpstat(1)		print information about the status	lpstat(1)
	uname(1)		print name of current UNIX system	uname(1)
	news(1)		print news items	news(1)
send/cancel requests to an LP			print service lp(1) cancel(1)	lp(1)
about the status of the LP			print service /print information	lpstat(1)
	jwin(1)		print size of layer	jwin(1)
	id(1M)		print user and group IDs and names	id(1M)
disable(1)	enable/disable LP		printers enable(1)	enable(1)
nice(1)	run a command at low		priority	nice(1)
timex(1)	time a command; report		process data and system activity	timex(1)
	kill(1)	terminate a	process	kill(1)
		ps(1)	report process status	ps(1)
	wait(1)	await completion of	process	wait(1)
	awk(1)	pattern scanning and	processing language	awk(1)
	nawk(1)	pattern scanning and	processing language	nawk(1)
	mailx(1)	interactive message	processing system	mailx(1)
/u3b(1)	u3b2(1)	u3b5(1)	processor type truth value	machid(1)
ASSIST	menus or command/	astgen(1)	program for generating/modifying	astgen(1)
	sdiff(1)	side-by-side difference	program	sdiff(1)
		units(1)	program	units(1)
the standard/restricted command			programming language /rsh(1) shell,	sh(1)
to commands and application			programs intro(1) introduction	intro(1)
	true(1)	false(1)	provide truth values	true(1)
			ps(1) report process status	ps(1)
copy	uuto(1C)	uupick(1C)	public UNIX-to-UNIX system file	uuto(1C)
			pwd(1) working directory name	pwd(1)
tput(1)	initialize a terminal or		query terminfo database	tput(1)
memory/	ipcrm(1)	remove a message	queue, semaphore set or shared	ipcrm(1)
run a command immune to hangups and			quits nohup(1)	nohup(1)
rmail(1)	send mail to users or		read mail mail(1)	mail(1)
		line(1)	read one line	line(1)
		ed(1)	red(1) text editor	ed(1)
a file for a pattern using full			regular expressions /search	egrep(1)
files	comm(1)	select or	reject lines common to two sorted	comm(1)
		join(1)	relational database operator	join(1)
			relogin(1M) rename login entry to	relogin(1M)
show current layer			reminder service	calendar(1)
	calendar(1)		remote terminal	ct(1C)
ct(1C)	spawn getty to a		remove a message queue, semaphore	ipcrm(1)
set or shared memory id	ipcrm(1)		remove files or directories	rm(1)
	rm(1)	rmdir(1)		

Permuted Index

constructs deroff(1) remove nroff/troff, tbl, and eqn deroff(1)
layer relogin(1M) rename login entry to show current relogin(1M)
uniqu(1) report repeated lines in a file uniqu(1)
facilities status ipcs(1) report inter-process communication ipcs(1)
and i-nodes df(1M) report number of free disk blocks df(1M)
activity timex(1) time a command; report process data and system timex(1)
ps(1) report process status ps(1)
uniqu(1) report repeated lines in a file uniqu(1)
sar(1) system activity reporter sar(1)
lp(1) cancel(1) send/cancel requests to an LP print service lp(1)
jterm(1) reset layer of windowing terminal jterm(1)
ismpx(1) return windowing terminal state ismpx(1)
col(1) filter reverse line-feeds col(1)
directories rm(1) rmdir(1) remove files or rm(1)
mail mail(1) rmail(1) send mail to users or read mail(1)
directories rm(1) rmdir(1) remove files or rm(1)
standard/restricted command/ sh(1) rsh(1) shell, the sh(1)
nice(1) run a command at low priority nice(1)
quits nohup(1) run a command immune to hangups and nohup(1)
sag(1G) system activity graph sag(1G)
sar(1) system activity reporter sar(1)
scanner bfs(1) big file scanning and processing language awk(1)
nawk(1) pattern scanning and processing language nawk(1)
editor based on ex vi(1) screen-oriented (visual) display vi(1)
program sdiff(1) side-by-side difference sdiff(1)
string fgrep(1) search a file for a character fgrep(1)
grep(1) search a file for a pattern grep(1)
full regular expressions egrep(1) search a file for a pattern using egrep(1)
sed(1) stream editor sed(1)
two sorted files comm(1) select or reject lines common to comm(1)
file cut(1) cut out selected fields of each line of a cut(1)
ipcrm(1) remove a message queue, semaphore set or shared memory id ipcrm(1)
mail(1) rmail(1) send mail to users or read mail mail(1)
service lp(1) cancel(1) send/cancel requests to an LP print lp(1)
calendar(1) reminder service calendar(1)
send/cancel requests to an LP print service lp(1) cancel(1) lp(1)
about the status of the LP print service /print information lpstat(1)
execution env(1) set environment for command env(1)
umask(1) set file-creation mode mask umask(1)
remove a message queue, semaphore set or shared memory id ipcrm(1) ipcrm(1)
tabs(1) set tabs on a terminal tabs(1)
date(1) print and set the date date(1)
stty(1) set the options for a terminal stty(1)
first user setup(1) initialize system for setup(1)
of/ paste(1) merge same lines of several files or subsequent lines paste(1)
standard/restricted command/ sh(1) rsh(1) shell, the sh(1)
a message queue, semaphore set or shared memory id ipcrm(1) remove ipcrm(1)
shl(1) shell layer manager shl(1)

command programming/ sh(1) rsh(1) shell, the standard/restricted sh(1)
 sh(1) shell layer manager shl(1)
 relogin(1M) rename login entry to
 sdiff(1) side-by-side difference program sdiff(1)
 login(1) sign on login(1)
 jwin(1) print size of layer jwin(1)
 interval sleep(1) suspend execution for an sleep(1)
 sort(1) sort and/or merge files sort(1)
 sort(1) sort and/or merge files sort(1)
 sorted files comm(1) select comm(1)
 spawn getty to a remote terminal ct(1C)
 hashcheck(1) find spelling errors
 spelling/ spell(1) hashmake(1) spellin(1) spell(1)
 spellin(1) hashcheck(1) find spelling errors /hashmake(1) spell(1)
 split(1) split a file into pieces split(1)
 csplit(1) context split csplit(1)
 split(1) split a file into pieces split(1)
 standard/restricted command/ sh(1)
 status inquiry and job control uustat(1C)
 communication facilities status /report inter-process ipc(1)
 /print information about the status of the LP print service lpstat(1)
 ps(1) report process status ps(1)
 dsconfig(1) display data storage device configuration dsconfig(1)
 sed(1) stream editor sed(1)
 search a file for a character string fgrep(1) fgrep(1)
 terminal stty(1) set the options for a stty(1)
 user su(1M) become super-user or another su(1M)
 same lines of several files or subsequent lines of one file /merge paste(1)
 count of a file sum(1) print checksum and block sum(1)
 du(1M) summarize disk usage du(1M)
 sync(1M) update the super block sync(1M)
 su(1M) become super-user or another user su(1M)
 sleep(1) suspend execution for an interval sleep(1)
 sync(1M) update the super block sync(1M)
 system administration sysadm(1) menu interface to do sysadm(1)
 sag(1G) system activity graph sag(1G)
 sar(1) system activity reporter sar(1)
 a command; report process data and system activity timex(1) time timex(1)
 sysadm(1) menu interface to do system administration sysadm(1)
 uux(1C) UNIX-to-UNIX system command execution uux(1C)
 assist(1) assistance using UNIX system commands assist(1)
 uulog(1C) uuname(1C) UNIX-to-UNIX system copy uucp(1C) uucp(1C)
 cu(1C) call another UNIX system cu(1C)
 uupick(1C) public UNIX-to-UNIX system file copy uuto(1C) uuto(1C)
 setup(1) initialize system for first user setup(1)
 interactive message processing system mailx(1) mailx(1)
 uname(1) print name of current UNIX system uname(1)
 who(1) who is on the system who(1)
 tabs(1) set tabs on a terminal tabs(1)

Permuted Index

	tabs(1) set tabs on a terminal	tabs(1)
file	tail(1) deliver the last part of a	tail(1)
tar(1)	tape file archiver	tar(1)
	tar(1) tape file archiver	tar(1)
deroff(1) remove nroff/troff,	tbl, and eqn constructs	deroff(1)
	tee(1) pipe fitting	tee(1)
ct(1C) spawn getty to a remote	terminal	ct(1C)
jterm(1) reset layer of windowing	terminal	jterm(1)
tput(1) initialize a	terminal or query terminfo database	tput(1)
ismpx(1) return windowing	terminal state	ismpx(1)
stty(1) set the options for a	terminal	stty(1)
tabs(1) set tabs on a	terminal	tabs(1)
tty(1) get the name of the	terminal	tty(1)
layer multiplexor for windowing	terminals layers(1)	layers(1)
kill(1)	terminate a process	kill(1)
initialize a terminal or query	terminfo database tput(1)	tput(1)
command	test(1) condition evaluation	test(1)
ed(1) red(1)	text editor	ed(1)
ex(1)	text editor	ex(1)
casual users) edit(1)	text editor (variant of ex for	edit(1)
newform(1) change the format of a	text file	newform(1)
	time(1) time a command	time(1)
update access and modification	times of a file touch(1)	touch(1)
process data and system activity	timex(1) time a command; report	timex(1)
modification times of a file	touch(1) update access and	touch(1)
	tplot(1G) graphics filters	tplot(1G)
query terminfo database	tput(1) initialize a terminal or	tput(1)
	tr(1) translate characters	tr(1)
tr(1)	translate characters	tr(1)
values	true(1) false(1) provide truth	true(1)
u3b5(1) vax(1) get processor type	truth value /u3b(1) u3b2(1)	machid(1)
true(1) false(1) provide	truth values	true(1)
	tty(1) get the name of the terminal	tty(1)
file(1) determine file	type	file(1)
u3b5(1) vax(1) get processor	type truth value /u3b(1) u3b2(1)	machid(1)
processor type/ machid(1) pdp11(1)	u3b(1) u3b2(1) u3b5(1) vax(1) get	machid(1)
machid(1) pdp11(1) u3b(1)	u3b2(1) u3b5(1) vax(1) get/	machid(1)
machid(1) pdp11(1) u3b(1) u3b2(1)	u3b5(1) vax(1) get processor type/	machid(1)
mask	umask(1) set file-creation mode	umask(1)
system	uname(1) print name of current UNIX	uname(1)
file	uniq(1) report repeated lines in a	uniq(1)
	units(1) conversion program	units(1)
assist(1) assistance using	UNIX system commands	assist(1)
cu(1C) call another	UNIX system	cu(1C)
uname(1) print name of current	UNIX system	uname(1)
execution uux(1C)	UNIX-to-UNIX system command	uux(1C)
uucp(1C) uulog(1C) uuname(1C)	UNIX-to-UNIX system copy	uucp(1C)
uuto(1C) uupick(1C) public	UNIX-to-UNIX system file copy	uuto(1C)
pack(1) pcat(1)	unpack(1) compress and expand files	pack(1)

times of a file touch(1)
 sync(1M) update access and modification touch(1)
 du(1M) summarize disk update the super block sync(1M)
 id(1M) print usage du(1M)
 crontab(1) user and group IDs and names id(1M)
 initialize system for first user crontab file crontab(1)
 su(1M) become super-user or another user setup(1) setup(1)
 write(1) write to another user su(1M)
 editor (variant of ex for casual users) edit(1) text edit(1)
 mail(1) rmail(1) send mail to users or read mail mail(1)
 wall(1) write to all users wall(1)
 /search a file for a pattern using full regular expressions egrep(1)
 assist(1) assistance using UNIX system commands assist(1)
 uustat(1C) uucp status inquiry and job control uustat(1C)
 UNIX-to-UNIX system copy uucp(1C) uulog(1C) uuname(1C) uucp(1C)
 system copy uucp(1C) uulog(1C) uuname(1C) UNIX-to-UNIX uucp(1C)
 uucp(1C) uulog(1C) uuname(1C) UNIX-to-UNIX system copy uucp(1C)
 system file copy uuto(1C) uupick(1C) public UNIX-to-UNIX uuto(1C)
 job control uustat(1C) uucp status inquiry and uustat(1C)
 UNIX-to-UNIX system file copy uuto(1C) uupick(1C) public uuto(1C)
 execution uux(1C) UNIX-to-UNIX system command uux(1C)
 vax(1) get processor type truth value /u3b(1) u3b2(1) u3b5(1) machid(1)
 true(1) false(1) provide truth values true(1)
 edit(1) text editor (variant of ex for casual users) edit(1)
 /pdp11(1) u3b(1) u3b2(1) u3b5(1) vax(1) get processor type truth / machid(1)
 display editor based on ex vi(1) screen-oriented (visual) vi(1)
 vi(1) screen-oriented (visual) display editor based on ex vi(1)
 wait(1) await completion of process wait(1)
 wall(1) write to all users wall(1)
 wc(1) word count wc(1)
 who(1) who is on the system who(1)
 jterm(1) reset layer of windowing terminal jterm(1)
 ismpx(1) return windowing terminal state ismpx(1)
 layers(1) layer multiplexor for windowing terminals layers(1)
 wc(1) word count wc(1)
 cd(1) change working directory cd(1)
 pwd(1) working directory name pwd(1)
 wall(1) write to all users wall(1)
 write(1) write to another user write(1)
 write(1) write to another user write(1)
 and execute command xargs(1) construct argument list(s) xargs(1)



NAME

intro – introduction to commands and application programs

DESCRIPTION

This section describes, in alphabetical order, commands available for the AT&T 3B2 Computer. Certain distinctions of purpose are made in the headings.

The following Utility packages are delivered with the computer:

- AT&T Windowing Utilities
- Basic Networking Utilities
- Cartridge Tape Controller Utilities
- Directory and File Management Utilities
- Editing Utilities
- Essential Utilities
- Form and Menu Language Interpreter Utilities
- Framed Access Command Environment Utilities
- Inter-process Communications Utilities
- Line Printer Spooling Utilities
- Security Administration Utilities
- Spell Utilities
- Terminal Information Utilities
- User Environment Utilities

The following Utility Packages are available for purchase:

- 2K File System Utilities
- ASSIST Utilities
- Networking Support Utilities
- Remote File Sharing Utilities
- System Performance Analysis Utilities

Manual Page Command Syntax

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

name [-*option*...] [*cmdarg*...]

where:

- [] Surround an *option* or *cmdarg* that is not required.
- ... Indicates multiple occurrences of the *option* or *cmdarg*.
- name* The name of an executable file.
- option* (Always preceded by a “-”.)
noargletter ... or,
argletter optarg[...]
- noargletter* A single letter representing an option without an option-argument. Note that more than one *noargletter* option can be grouped after one “-” (Rule 5, below).
- argletter* A single letter representing an option requiring an option-argument.

- optarg* An option-argument (character string) satisfying a preceding *argletter*. Note that groups of *optargs* following an *argletter* must be separated by commas, or separated by white space and quoted (Rule 8, below).
- cmdarg* Path name (or other command argument) *not* beginning with "--", or "-" by itself indicating the standard input.

Command Syntax Standard: Rules

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

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

SEE ALSO

getopts(1),
exit(2), *wait(2)*, *getopt(3C)* in the *Programmer's Reference Manual*.
How to Get Started, at the front of this document.

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of "normal" termination) one supplied by the program [see *wait(2)* and *exit(2)*]. The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and non-zero to indicate troubles such as erroneous parameters, or bad or inaccessible data. It is called variously "exit code", "exit status", or "return code", and is described only where special conventions are involved.

WARNINGS

Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings and therefore become confused upon encountering a null character (the string terminator) within a line.



NAME

`ar` – archive and library maintainer for portable archives

SYNOPSIS

`ar key [posname] afile [name] ...`

DESCRIPTION

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

When `ar` creates an archive, it creates headers in a format that is portable across all machines. The portable archive format and structure is described in detail in `ar(4)`. The archive symbol table [described in `ar(4)`] is used by the link editor [`ld(1)`] to effect multiple passes over libraries of object files in an efficient manner. An archive symbol table is only created and maintained by `ar` when there is at least one object file in the archive. The archive symbol table is in a specially named file which is always the first file in the archive. This file is never mentioned or accessible to the user. Whenever the `ar(1)` command is used to create or update the contents of such an archive, the symbol table is rebuilt. The `s` option described below will force the symbol table to be rebuilt.

Unlike command options, the command key is a required part of `ar`'s command line. The key (which may begin with a `-`) is formed with one of the following letters: **drqtpmx**. Arguments to the key, alternatively, are made with one of more of the following set: **vuaibcls**. *Posname* is an archive member name used as a reference point in positioning other files in the archive. *Afile* is the archive file. The *names* are constituent files in the archive file. The meanings of the key characters are as follows:

- d** Delete the named files from the archive file.
- r** Replace the named files in the archive file. If the optional character **u** is used with **r**, then only those files with dates of modification later than the archive files are replaced. If an optional positioning character from the set **abi** is used, then the *posname* argument must be present and specifies that new files are to be placed after (**a**) or before (**b** or **i**) *posname*. Otherwise new files are placed at the end.
- q** Quickly append the named files to the end of the archive file. Optional positioning characters are invalid. The command does not check whether the added members are already in the archive. This option is useful to avoid quadratic behavior when creating a large archive piece-by-piece. Unchecked, the file may grow exponentially up to the second degree.
- t** Print a table of contents of the archive file. If no names are given, all files in the archive are tabled. If names are given, only those files are tabled.
- p** Print the named files in the archive.

- m** Move the named files to the end of the archive. If a positioning character is present, then the *posname* argument must be present and, as in *r*, specifies where the files are to be moved.
- x** Extract the named files. If no names are given, all files in the archive are extracted. In neither case does *x* alter the archive file.

The meanings of the key arguments are as follows:

- v** Give a verbose file-by-file description of the making of a new archive file from the old archive and the constituent files. When used with *t*, give a long listing of all information about the files. When used with *x*, precede each file with a name.
- c** Suppress the message that is produced by default when *afile* is created.
- l** Place temporary files in the local (current working) directory rather than in the default temporary directory, *TMPDIR*.
- s** Force the regeneration of the archive symbol table even if *ar(1)* is not invoked with a command which will modify the archive contents. This command is useful to restore the archive symbol table after the *strip(1)* command has been used on the archive.

FILES

\$TMPDIR/* temporary files

\$TMPDIR is usually */usr/tmp* but can be redefined by setting the environment variable *TMPDIR* [see *tempnam()* in *tempnam(3S)*].

SEE ALSO

ld(1), *lorder(1)*, *strip(1)*, *tempnam(3S)* in the *Programmer's Reference Manual*.
a.out(4), *ar(4)* in the *System Administrator's Reference Manual*.

NOTES

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

NAME

assist — assistance using UNIX system commands

SYNOPSIS

```
assist [name]
assist [-s]
assist [-c name]
```

DESCRIPTION

The *assist* command invokes the ASSIST menu interface software for the UNIX system. The ASSIST menus categorize UNIX system commands according to function in a hierarchy. The menus lead to full-screen forms (called command forms) that aid in the execution of a syntactically correct UNIX system command line. The menus also lead to interactive simulations of UNIX system commands or concepts (called walkthrus).

If you type *assist* without options, you enter at the top of the menu interface hierarchy. New users may wish to use the *-s* option to select an introductory tutorial explaining how to use the ASSIST software. Options are:

```
name          invoke an ASSIST-supported UNIX system command form or walk-
                thru for name
-c name       invoke the version of name that is in your current directory
-s            reinvoke the ASSIST setup module and check or modify your ter-
                minal variable; or access the introductory information about
                ASSIST
```

When you invoke *assist*, you perform operations within the program by using *assist* commands. To see a list of the *assist* commands, type *^A* (control-*a*) or *f8* (function-key 8) when you are in *assist*. When you do this, a list of the commands is printed on the terminal screen. The entire set of commands is described in the "Glossary of ASSIST Commands" in the *ASSIST Software User's Guide*.

EXAMPLE

This example illustrates how to invoke a particular command form directly. In this case, *mkdir* is the desired command form.

```
assist mkdir
```

FILES

```
$HOME/.assistrc    information needed by assist (for example, about the
                    terminal you are using)
/usr/lib/assist   default directory containing assist command forms,
                    walkthrus, and executable programs
/usr/lib/readme/assist basic information about installing assist and astgen
```

NOTES

The first time you invoke *assist* it will ignore any options you give and ask for information about the terminal you are using. Once it has saved this information in a file named *.assistrc* in your home directory, it will show you a list of basic *assist* commands and offer you an introduction to ASSIST.

ASSIST(1)

(ASSIST Utilities)

ASSIST(1)

SEE ALSO

astgen(1).

ASSIST Software User's Guide.

`/usr/lib/readme/assist.`

NAME

astgen – program for generating/modifying ASSIST menus or command forms

SYNOPSIS

astgen name[.fs]

DESCRIPTION

astgen is an interactive program to generate information files (ASCII text data files) that define a menu or command form used by the *assist(1)* program.

Both the *astgen* and *assist(1)* programs recognize and process information files whose names are suffixed with three characters: *.fs*. If no *.fs* file exists for the specified name, *astgen* assumes that a new menu or command form is to be created. If *name* is given without *.fs*, *astgen* automatically will create the file: *name.fs*.

Details of how to use *astgen* are given in the *ASSIST Software Development Tools Guide*.

SEE ALSO

assist(1).
ASSIST Software Development Tools Guide.
ASSIST Software User's Guide.



NAME

at, *batch* – execute commands at a later time

SYNOPSIS

at *time* [*date*] [+ *increment*]

at -*r* *job*...

at -*l* [*job* ...]

batch

DESCRIPTION

at and *batch* read commands from standard input to be executed at a later time. *at* allows you to specify when the commands should be executed, while jobs queued with *batch* will execute when system load level permits. *at* may be used with the following options:

-*r* Removes jobs previously scheduled with *at*.

-*l* Reports all jobs scheduled for the invoking user.

Standard output and standard error output are mailed to the user unless they are redirected elsewhere. The shell environment variables, current directory, *umask*, and *ulimit* are retained when the commands are executed. Open file descriptors, traps, and priority are lost.

Users are permitted to use *at* if their name appears in the file */usr/lib/cron/at.allow*. If that file does not exist, the file */usr/lib/cron/at.deny* is checked to determine if the user should be denied access to *at*. If neither file exists, only root is allowed to submit a job. If *at.deny* is empty, global usage is permitted. The allow/deny files consist of one user name per line. These files can only be modified by the superuser.

The *time* may be specified as 1, 2, or 4 digits. One and two digit numbers are taken to be hours, four digits to be hours and minutes. The time may alternately be specified as two numbers separated by a colon, meaning *hour:minute*. A suffix *am* or *pm* may be appended; otherwise a 24-hour clock time is understood. The suffix *zulu* may be used to indicate GMT. The special names *noon*, *midnight*, *now*, and *next* are also recognized.

An optional *date* may be specified as either a month name followed by a day number (and possibly year number preceded by an optional comma) or a day of the week (fully spelled or abbreviated to three characters). Two special "days", *today* and *tomorrow* are recognized. If no *date* is given, *today* is assumed if the given hour is greater than the current hour and *tomorrow* is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.

The optional *increment* is simply a number suffixed by one of the following: *minutes*, *hours*, *days*, *weeks*, *months*, or *years*. (The singular form is also accepted.)

Thus legitimate commands include:

```
at 0815am Jan 24
at 8:15am Jan 24
at now + 1 day
at 5 pm Friday
```

at and *batch* write the job number and schedule time to standard error.

batch submits a batch job. It is almost equivalent to "at now", but not quite. For one, it goes into a different queue. For another, "at now" will respond with the error message **too late**.

at -r removes jobs previously scheduled by *at* or *batch*. The job number is the number given to you previously by the *at* or *batch* command. You can also get job numbers by typing *at -l*. You can only remove your own jobs unless you are the super-user.

EXAMPLES

The *at* and *batch* commands read from standard input the commands to be executed at a later time. *sh(1)* provides different ways of specifying standard input. Within your commands, it may be useful to redirect standard output.

This sequence can be used at a terminal:

```
batch
sort filename >outfile
<control-D> (hold down 'control' and depress 'D')
```

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

```
batch <<!
sort filename 2>&1 >outfile | mail loginid
!
```

To have a job reschedule itself, invoke *at* from within the shell procedure, by including code similar to the following within the shell file:

```
echo "sh shellfile" | at 1900 thursday next week
```

FILES

/usr/lib/cron	main cron directory
/usr/lib/cron/at.allow	list of allowed users
/usr/lib/cron/at.deny	list of denied users
/usr/lib/cron/queue	scheduling information
/usr/spool/cron/atjobs	spool area

SEE ALSO

kill(1), mail(1), nice(1), ps(1), sh(1), sort(1).
cron(1M) in the *System Administrator's Reference Manual*.

DIAGNOSTICS

Complains about various syntax errors and times out of range.

NAME

awk – pattern scanning and processing language

SYNOPSIS

```
awk [ -Fc ] [ prog ] [ parameters ] [ files ]
```

DESCRIPTION

awk scans each input *file* for lines that match any of a set of patterns specified in *prog*. With each pattern in *prog* there can be an associated action that will be performed when a line of a *file* matches the pattern. The set of patterns may appear literally as *prog*, or in a file specified as *-f file*. The *prog* string should be enclosed in single quotes (') to protect it from the shell.

Parameters, in the form *x=... y=...* etc., may be passed to *awk*.

Files are read in order; if there are no files, the standard input is read. The file name *-* means the standard input. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

An input line is made up of fields separated by white space. (This default can be changed by using *FS*; see below). The fields are denoted *\$1*, *\$2*, ...; *\$0* refers to the entire line.

A pattern-action statement has the form:

```
pattern { action }
```

A missing action means print the line; a missing pattern always matches. An action is a sequence of statements. A statement can be one of the following:

```
if ( conditional ) statement [ else statement ]
while ( conditional ) statement
for ( expression ; conditional ; expression ) statement
break
continue
{ [ statement ] ... }
variable = expression
print [ expression-list ] [ >expression ]
printf format [ , expression-list ] [ >expression ]
next # skip remaining patterns on this input line
exit # skip the rest of the input
```

Statements are terminated by semicolons, new-lines, or right braces. An empty expression-list stands for the whole line. Expressions take on string or numeric values as appropriate, and are built using the operators *+*, *-*, ***, */*, *%*, and concatenation (indicated by a blank). The C operators *++*, *--*, *+=*, *-=*, **=*, */=*, and *%=* are also available in expressions. Variables may be scalars, array elements (denoted *x[i]*) or fields. Variables are initialized to the null string. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. String constants are quoted (").

The *print* statement prints its arguments on the standard output (or on a file if *>expr* is present), separated by the current output field separator, and terminated by the output record separator. The *printf* statement formats its expression list according to the format [see *printf(3S)* in the *Programmer's Reference Manual*].

The built-in function *length* returns the length of its argument taken as a string, or of the whole line if no argument. There are also built-in functions *exp*, *log*, *sqrt*, and *int*. The last truncates its argument to an integer; *substr(s, m, n)* returns the *n*-character substring of *s* that begins at position *m*. The function *sprintf(fmt, expr, expr, ...)* formats the expressions according to the *printf(3S)* format given by *fmt* and returns the resulting string.

Patterns are arbitrary Boolean combinations (*!*, *||*, *&&*, and parentheses) of regular expressions and relational expressions. Regular expressions must be surrounded by slashes and are as in *egrep* (see *grep(1)*). Isolated regular expressions in a pattern apply to the entire line. Regular expressions may also occur in relational expressions. A pattern may consist of two patterns separated by a comma; in this case, the action is performed for all lines between an occurrence of the first pattern and the next occurrence of the second.

A relational expression is one of the following:

```
expression matchop regular-expression
expression relop expression
```

where a *relop* is any of the six relational operators in C, and a *matchop* is either *~* (for *contains*) or *!~* (for *does not contain*). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of these.

The special patterns BEGIN and END may be used to capture control before the first input line is read and after the last. BEGIN must be the first pattern, END the last.

A single character *c* may be used to separate the fields by starting the program with:

```
BEGIN { FS = c }
```

or by using the *-Fc* option.

Other variable names with special meanings include *NF*, the number of fields in the current record; *NR*, the ordinal number of the current record; *FILENAME*, the name of the current input file; *OFS*, the output field separator (default blank); *ORS*, the output record separator (default new-line); and *OFMT*, the output format for numbers (default *%.6g*).

EXAMPLES

Print lines longer than 72 characters:

```
length > 72
```

Print first two fields in opposite order:

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

Add up first column, print sum and average:

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

Print fields in reverse order:

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

Print all lines between start/stop pairs:

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

Print all lines whose first field is different from previous one:

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

Print file, filling in page numbers starting at 5:

```
    /Page/ { $2 = n++; }
    { print }
```

command line: `awk -f program n=5 input`

SEE ALSO

`grep(1)`, `sed(1)`.

`lex(1)`, `printf(3S)` in the *Programmer's Reference Manual*.

BUGS

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

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



NAME

banner – make posters

SYNOPSIS

banner strings

DESCRIPTION

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

SEE ALSO

echo(1).



NAME

basename, dirname – deliver portions of path names

SYNOPSIS

```
basename string [ suffix ]  
dirname string
```

DESCRIPTION

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

Dirname delivers all but the last level of the path name in *string*.

EXAMPLES

The following example, invoked with the argument `/usr/src/cmd/cat.c`, compiles the named file and moves the output to a file named `cat` in the current directory:

```
cc $1  
mv a.out `basename $1 \.c`
```

The following example will set the shell variable `NAME` to `/usr/src/cmd`:

```
NAME=`dirname /usr/src/cmd/cat.c`
```

SEE ALSO

sh(1).



NAME

bc – arbitrary-precision arithmetic language

SYNOPSIS

bc [*-c*] [*-l*] [*file ...*]

DESCRIPTION

bc is an interactive processor for a language that resembles C but provides unlimited precision arithmetic. It takes input from any files given, then reads the standard input. The *bc*(1) utility is actually a preprocessor for *dc*(1), which it invokes automatically unless the *-c* option is present. In this case the *dc* input is sent to the standard output instead. The options are as follows:

-c Compile only. The output is sent to the standard output.

-l Argument stands for the name of an arbitrary precision math library.

The syntax for *bc* programs is as follows; L means letter a–z, E means expression, S means statement.

Comments

are enclosed in */** and **/*.

Names

simple variables: L

array elements: L [E]

The words “ibase”, “obase”, and “scale”

Other operands

arbitrarily long numbers with optional sign and decimal point.

(E)

sqrt (E)

length (E) number of significant decimal digits

scale (E) number of digits right of decimal point

L (E , ... , E)

Operators

+ - * / % ^ (% is remainder; ^ is power)

++ -- (prefix and postfix; apply to names)

-- <= >= != < >

- -+ -- -* -/ -% -^

Statements

E

{ S ; ... ; S }

if (E) S

while (E) S

for (E ; E ; E) S

null statement

break

quit

Function definitions

define L (L , ... , L) {

auto L , ... , L

S ; ... S

```

    return ( E )
}

```

Functions in `-l` math library

```

s(x)   sine
c(x)   cosine
e(x)   exponential
l(x)   log
a(x)   arctangent
j(n,x) Bessel function

```

All function arguments are passed by value.

The value of a statement that is an expression is printed unless the main operator is an assignment. Either semicolons or new-lines may separate statements. Assignment to *scale* influences the number of digits to be retained on arithmetic operations in the manner of *dc(1)*. Assignments to *ibase* or *obase* set the input and output number radix respectively.

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

EXAMPLE

```

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

```

defines a function to compute an approximate value of the exponential function and

```

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

```

prints approximate values of the exponential function of the first ten integers.

FILES

```

/usr/lib/lib.b  mathematical library
/usr/bin/dc    desk calculator proper

```

SEE ALSO

`dc(1)`.

BUGS

The *bc* command does not yet recognize the logical operators, *&&* and *||*.
For statement must have all three expressions (E's).
Quit is interpreted when read, not when executed.



NAME

`bdiff` – big diff

SYNOPSIS

`bdiff file1 file2 [n] [-s]`

DESCRIPTION

bdiff is used in a manner analogous to *diff*(1) to find which lines in two files must be changed to bring the files into agreement. Its purpose is to allow processing of files which are too large for *diff*.

The parameters to *bdiff* are:

file1 (*file2*)

The name of a file to be used. If *file1* (*file2*) is `-`, the standard input is read.

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

`-s` Specifies that no diagnostics are to be printed by *bdiff* (silent option). Note, however, that this does not suppress possible diagnostic messages from *diff*(1), which *bdiff* calls.

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

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

FILES

`/tmp/bd?????`

SEE ALSO

diff(1), *help*(1).

DIAGNOSTICS

Use *help*(1) for explanations.



NAME

bfs – big file scanner

SYNOPSIS

bfs [-] name

DESCRIPTION

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

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

All address expressions described under *ed*(1) are supported. In addition, regular expressions may be surrounded with two symbols besides */* and *?*: *>* indicates downward search without wrap-around, and *<* indicates upward search without wrap-around. There is a slight difference in mark names: only the letters *a* through *z* may be used, and all 26 marks are remembered.

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

xf *file*

Further commands are taken from the named *file*. When an end-of-file is reached, an interrupt signal is received or an error occurs, reading resumes with the file containing the *xf*. The *xf* commands may be nested to a depth of 10.

xn List the marks currently in use (marks are set by the *k* command).

xo [*file*]

Further output from the *p* and null commands is diverted to the named *file*, which, if necessary, is created mode 666 (readable and writable by everyone), unless your *umask* setting (see *umask*(1)) dictates otherwise. If *file* is missing, output is diverted to the standard output. Note that each diversion causes truncation or creation of the file.

: *label*

This positions a *label* in a command file. The *label* is terminated by new-line, and blanks between the *:* and the start of the *label* are

ignored. This command may also be used to insert comments into a command file, since labels need not be referenced.

(. . .) **xb**/*regular expression/label*

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

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

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

xb/^/ *label*

is an unconditional jump.

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

xt *number*

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

xv[*digit*][*spaces*][*value*]

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

```
1,%5p
1,%5
%6
```

will all print the first 100 lines.

```
g/%5/p
```

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

```
g/".*\%[cds]/p
```

could be used to match and list lines containing *printf* of characters, decimal integers, or strings.

Another feature of the **xv** command is that the first line of output from a UNIX system command can be stored into a variable. The

only requirement is that the first character of *value* be an *!*. For example:

```
.w junk
xv5!cat junk
!rm junk
!echo "%5"
xv6!expr %6 + 1
```

would put the current line into variable *5*, print it, and increment the variable *6* by one. To escape the special meaning of *!* as the first character of *value*, precede it with a **.

```
xv7!\date
```

stores the value *!date* into variable *7*.

xbz *label*

xbn *label*

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

```
xv55
: 1
/size/
xv5!expr %5 - 1
!if 0%5 != 0 exit 2
xbn 1
xv45
: 1
/size/
xv4!expr %4 - 1
!if 0%4 = 0 exit 2
xbz 1
```

xc [*switch*]

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

SEE ALSO

csplit(1), *ed(1)*, *umask(1)*.

DIAGNOSTICS

? for errors in commands, if prompting is turned off. Self-explanatory error messages when prompting is on.



NAME

cal – print calendar

SYNOPSIS

cal [[month] year]

DESCRIPTION

cal prints a calendar for the specified year. If a month is also specified, a calendar just for that month is printed. If neither is specified, a calendar for the present month is printed. *Year* can be between 1 and 9999. The *month* is a number between 1 and 12. The calendar produced is that for England and the United States.

EXAMPLES

An unusual calendar is printed for September 1752. That is the month 11 days were skipped to make up for lack of leap year adjustments. To see this calendar, type: **cal 9 1752**

BUGS

The year is always considered to start in January even though this is historically naive.

Beware that "cal 83" refers to the early Christian era, not the 20th century.



NAME

calendar — reminder service

SYNOPSIS

calendar [-]

DESCRIPTION

calendar consults the file **calendar** in the current directory and prints out lines that contain today's or tomorrow's date anywhere in the line. Most reasonable month-day dates such as "Aug. 24," "august 24," "8/24," etc., are recognized, but not "24 August" or "24/8". On weekends "tomorrow" extends through Monday.

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

FILES

/usr/lib/calprog to figure out today's and tomorrow's dates
/etc/passwd
/tmp/cal*

SEE ALSO

mail(1).

BUGS

Your calendar must be public information for you to get reminder service. *calendar's* extended idea of "tomorrow" does not account for holidays.



NAME

`cat` – concatenate and print files

SYNOPSIS

`cat [-u] [-s] [-v [-t] [-e]] file ...`

DESCRIPTION

`cat` reads each *file* in sequence and writes it on the standard output. Thus:

```
cat file
```

prints **file** on your terminal, and:

```
cat file1 file2 >file3
```

concatenates **file1** and **file2**, and writes the results in **file3**.

If no input file is given, or if the argument `-` is encountered, `cat` reads from the standard input file.

The following options apply to `cat`:

- `-u` The output is not buffered. (The default is buffered output.)
- `-s` `cat` is silent about non-existent files.
- `-v` Causes non-printing characters (with the exception of tabs, new-lines and form-feeds) to be printed visibly. ASCII control characters (octal 000 - 037) are printed as `^n`, where *n* is the corresponding ASCII character in the range octal 100 - 137 (`@`, `A`, `B`, `C`, . . . , `X`, `Y`, `Z`, `[`, `\`, `]`, `^`, and `_`); the DEL character (octal 0177) is printed `^?`. Other non-printable characters are printed as `M-x`, where *x* is the ASCII character specified by the low-order seven bits.

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

- `-t` Causes tabs to be printed as `^I`'s and formfeeds to be printed as `^L`'s.
- `-e` Causes a `$` character to be printed at the end of each line (prior to the new-line).

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

WARNING

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

```
cat file1 file2 >file1
```

will cause the original data in **file1** to be lost.

SEE ALSO

`cp(1)`, `pg(1)`, `pr(1)`.



NAME

cd – change working directory

SYNOPSIS

cd [directory]

DESCRIPTION

If *directory* is not specified, the value of shell parameter \$HOME is used as the new working directory. If *directory* specifies a complete path starting with /, .., .., *directory* becomes the new working directory. If neither case applies, *cd* tries to find the designated directory relative to one of the paths specified by the \$CDPATH shell variable. \$CDPATH has the same syntax as, and similar semantics to, the \$PATH shell variable. *cd* must have execute (search) permission in *directory*.

Because a new process is created to execute each command, *cd* would be ineffective if it were written as a normal command; therefore, it is recognized and is internal to the shell.

SEE ALSO

pwd(1), sh(1).
chdir(2) in the *Programmer's Reference Manual*.



NAME

chmod – change mode

SYNOPSIS

chmod mode file ...

chmod mode directory ...

DESCRIPTION

The permissions of the named *files* or *directories* are changed according to **mode**, which may be symbolic or absolute. Absolute changes to permissions are stated using octal numbers:

```
chmod nnn file(s)
```

where *n* is a number from 0 to 7. Symbolic changes are stated using mnemonic characters:

```
chmod a operator b file(s)
```

where *a* is one or more characters corresponding to **user**, **group**, or **other**; where *operator* is +, -, and =, signifying assignment of permissions; and where *b* is one or more characters corresponding to type of permission.

An absolute mode is given as an octal number constructed from the OR of the following modes:

```
4000    set user ID on execution
20#0    set group ID on execution if # is 7, 5, 3, or 1
         enable mandatory locking if # is 6, 4, 2, or 0
1000    sticky bit is turned on ((see chmod(2))
0400    read by owner
0200    write by owner
0100    execute (search in directory) by owner
0070    read, write, execute (search) by group
0007    read, write, execute (search) by others
```

Symbolic changes are stated using letters that correspond both to access classes and to the individual permissions themselves. Permissions to a file may vary depending on your user identification number (UID) or group identification number (GID). Permissions are described in three sequences each having three characters:

```
User  Group  Other
```

```
rwX   rwX   rwX
```

This example (meaning that **user**, **group**, and **others** all have reading, writing, and execution permission to a given file) demonstrates two categories for granting permissions: the access class and the permissions themselves.

Thus, to change the mode of a file's (or directory's) permissions using *chmod*'s symbolic method, use the following syntax for mode:

```
[ who ] operator [ permission(s) ], ...
```

A command line using the symbolic method would appear as follows:

```
chmod g+rw file
```

This command would make *file* readable and writable by the group.

The *who* part can be stated as one or more of the following letters:

u	user's permissions
g	group's permissions
o	others permissions

The letter **a** (all) is equivalent to **ugo** and is the default if *who* is omitted.

Operator can be **+** to add *permission* to the file's mode, **-** to take away *permission*, or **=** to assign *permission* absolutely. (Unlike other symbolic operations, **=** has an absolute effect in that it resets all other bits.) Omitting *permission* is only useful with **=** to take away all permissions.

Permission is any compatible combination of the following letters:

r	reading permission
w	writing permission
x	execution permission
s	user or group set-ID is turned on
t	sticky bit is turned on
l	mandatory locking will occur during access

Multiple symbolic modes separated by commas may be given, though no spaces may intervene between these modes. Operations are performed in the order given. Multiple symbolic letters following a single operator cause the corresponding operations to be performed simultaneously. The letter **s** is only meaningful with **u** or **g**, and **t** only works with **u**.

Mandatory file and record locking (**l**) refers to a file's ability to have its reading or writing permissions locked while a program is accessing that file. It is not possible to permit group execution and enable a file to be locked on execution at the same time. In addition, it is not possible to turn on the set-group-ID and enable a file to be locked on execution at the same time. The following examples,

```
chmod g+x,+l file
```

```
chmod g+s,+l file
```

are, therefore, illegal usages and will elicit error messages.

Only the owner of a file or directory (or the super-user) may change a file's mode. Only the super-user may set the sticky bit on a non-directory file. If you are not super-user, **chmod** will mask the sticky-bit but will not return an error. In order to turn on a file's set-group-ID, your own group ID must correspond to the file's and group execution must be set.

EXAMPLES

```
chmod a-x file
```

```
chmod 444 file
```

The first examples deny execution permission to all. The absolute (octal) example permits only reading permissions.

```
chmod go+rw file
```

```
chmod 066 file
```

These examples make a file readable and writable by the group and others.

```
chmod +l file
```

This causes a file to be locked during access.

```
chmod -rwx,g+s file
```

```
chmod 2777 file
```

These last two examples enable all to read, write, and execute the file; and they turn on the set group-ID.

NOTES

In a Remote File Sharing environment, you may not have the permissions that the output of the `ls -l` command leads you to believe. For more information see the "Mapping Remote Users" section of Chapter 10 of the *System Administrator's Guide*.

SEE ALSO

`ls(1)`.

`chmod(2)` in the *Programmer's Reference Manual*.



NAME

chown, chgrp – change owner or group

SYNOPSIS

chown owner file ...

chown owner directory ...

chgrp group file ...

chgrp group directory ...

DESCRIPTION

chown changes the owner of the *files* or *directories* to *owner*. The owner may be either a decimal user ID or a login name found in the password file.

chgrp changes the group ID of the *files* or *directories* to *group*. The group may be either a decimal group ID or a group name found in the group file.

If either command is invoked by other than the super-user, the set-user-ID and set-group-ID bits of the file mode, 04000 and 02000 respectively, will be cleared.

Only the owner of a file (or the super-user) may change the owner or group of that file.

FILES

/etc/passwd

/etc/group

NOTES

In a Remote File Sharing environment, you may not have the permissions that the output of the `ls -l` command leads you to believe. For more information see the "Mapping Remote Users" section of Chapter 10 of the *System Administrator's Guide*.

SEE ALSO

`chmod(1)`.

`chown(2)` in the *Programmer's Reference Manual*.

`group(4)`, `passwd(4)` in the *System Administrator's Reference Manual*.



NAME

`cmp` – compare two files

SYNOPSIS

`cmp [-l] [-s] file1 file2`

DESCRIPTION

The two files are compared. (If *file1* is `-`, the standard input is used.) Under default options, *cmp* makes no comment if the files are the same; if they differ, it announces the byte and line number at which the difference occurred. If one file is an initial subsequence of the other, that fact is noted.

Options:

- `-l` Print the byte number (decimal) and the differing bytes (octal) for each difference.
- `-s` Print nothing for differing files; return codes only.

SEE ALSO

`comm(1)`, `diff(1)`.

DIAGNOSTICS

Exit code 0 is returned for identical files, 1 for different files, and 2 for an inaccessible or missing argument.



NAME

`col` - filter reverse line-feeds

SYNOPSIS

`col [-b] [-f] [-x] [-p]`

DESCRIPTION

`col` reads from the standard input and writes onto the standard output. It performs the line overlays implied by reverse line feeds (ASCII code ESC-7), and by forward and reverse half-line-feeds (ESC-9 and ESC-8). `col` is particularly useful for filtering multicolumn output made with the `.rt` command of `nroff` and output resulting from use of the `tbl(1)` preprocessor.

If the `-b` option is given, `col` assumes that the output device in use is not capable of backspacing. In this case, if two or more characters are to appear in the same place, only the last one read will be output.

Although `col` accepts half-line motions in its input, it normally does not emit them on output. Instead, text that would appear between lines is moved to the next lower full-line boundary. This treatment can be suppressed by the `-f` (fine) option; in this case, the output from `col` may contain forward half-line-feeds (ESC-9), but will still never contain either kind of reverse line motion.

Unless the `-x` option is given, `col` will convert white space to tabs on output wherever possible to shorten printing time.

The ASCII control characters SO (\017) and SI (\016) are assumed by `col` to start and end text in an alternate character set. The character set to which each input character belongs is remembered, and on output SI and SO characters are generated as appropriate to ensure that each character is printed in the correct character set.

On input, the only control characters accepted are space, backspace, tab, return, new-line, SI, SO, VT (\013), and ESC followed by 7, 8, or 9. The VT character is an alternate form of full reverse line-feed, included for compatibility with some earlier programs of this type. All other non-printing characters are ignored.

Normally, `col` will ignore any escape sequences unknown to it that are found in its input; the `-p` option may be used to cause `col` to output these sequences as regular characters, subject to overprinting from reverse line motions. The use of this option is highly discouraged unless the user is fully aware of the textual position of the escape sequences.

SEE ALSO

`nroff(1)`, `tbl(1)` in the *DOCUMENTER's WORKBENCH Software Release 2.0 Technical Discussion and Reference Manual*.

NOTES

The input format accepted by `col` matches the output produced by `nroff` with either the `-T37` or `-Tlp` options. Use `-T37` (and the `-f` option of `col`) if the ultimate disposition of the output of `col` will be a device that can interpret half-line motions, and `-Tlp` otherwise.

BUGS

Cannot back up more than 128 lines.

Allows at most 800 characters, including backspaces, on a line.

Local vertical motions that would result in backing up over the first line of the document are ignored. As a result, the first line must not have any superscripts.

NAME

`comm` - select or reject lines common to two sorted files

SYNOPSIS

`comm` [- [123]] file1 file2

DESCRIPTION

`comm` reads *file1* and *file2*, which should be ordered in ASCII collating sequence (see `sort(1)`), and produces a three-column output: lines only in *file1*; lines only in *file2*; and lines in both files. The file name `-` means the standard input.

Flags 1, 2, or 3 suppress printing of the corresponding column. Thus `comm -12` prints only the lines common to the two files; `comm -23` prints only lines in the first file but not in the second; `comm -123` prints nothing.

SEE ALSO

`cmp(1)`, `diff(1)`, `sort(1)`, `uniq(1)`.



NAME

cp, *ln*, *mv* – copy, link or move files

SYNOPSIS

```
cp file1 [ file2 ...] target
ln [ -f ] file1 [ file2 ...] target
mv [ -f ] file1 [ file2 ...] target
```

DESCRIPTION

file1 is copied (linked, moved) to *target*. Under no circumstance can *file1* and *target* be the same (take care when using *sh(1)* metacharacters). If *target* is a directory, then one or more files are copied (linked, moved) to that directory. If *target* is a file, its contents are destroyed.

If *mv* or *ln* determines that the mode of *target* forbids writing, it will print the mode (see *chmod(2)*), ask for a response, and read the standard input for one line; if the line begins with *y*, the *mv* or *ln* occurs, if permissible; if not, the command exits. For *mv*, when the parent directory of *file1* is writable and has the sticky bit set, one or more of the following conditions must be true:

- the user must own the file
- the user must own the directory
- the file must be writable by the user
- the user must be the super-user

When the *-f* option is used or if the standard input is not a terminal, no questions are asked and the *mv* or *ln* is done.

Only *mv* will allow *file1* to be a directory, in which case the directory rename will occur only if the two directories have the same parent; *file1* is renamed *target*. If *file1* is a file and *target* is a link to another file with links, the other links remain and *target* becomes a new file.

When using *cp*, if *target* is not a file, a new file is created which has the same mode as *file1* except that the sticky bit is not set unless you are super-user; the owner and group of *target* are those of the user. If *target* is a file, copying a file into *target* does not change its mode, owner, nor group. The last modification time of *target* (and last access time, if *target* did not exist) and the last access time of *file1* are set to the time the copy was made. If *target* is a link to a file, all links remain and the file is changed.

SEE ALSO

chmod(1), *cpio(1)*, *rm(1)*.

WARNINGS

ln will not link across file systems. This restriction is necessary because file systems can be added and removed.

BUGS

If *file1* and *target* lie on different file systems, *mv* must copy the file and delete the original. In this case any linking relationship with other files is lost.



NAME

`cpio` - copy file archives in and out

SYNOPSIS

`cpio -o[acBvV] [-C bufsize] [[-O file] [-M message]]`

`cpio -i[BcdmrtuvVfsSb6k] [-C bufsize] [[-I file] [-M message]] [pattern ...]`

`cpio -p[adlmuvV] directory`

DESCRIPTION

`cpio -o` (copy out) reads the standard input to obtain a list of path names and copies those files onto the standard output together with path name and status information. Output is padded to a 512-byte boundary by default.

`cpio -i` (copy in) extracts files from the standard input, which is assumed to be the product of a previous `cpio -o`. Only files with names that match *patterns* are selected. *patterns* are regular expressions given in the filename-generating notation of *sh*(1). In *patterns*, meta-characters `?`, `*`, and `[...]` match the slash (`/`) character, and backslash (`\`) is an escape character. A `!` meta-character means *not*. (For example, the `!abc*` pattern would exclude all files that begin with `abc`.) Multiple *patterns* may be specified and if no *patterns* are specified, the default for *patterns* is `*` (i.e., select all files). Each *pattern* must be enclosed in double quotes otherwise the name of a file in the current directory is used. Extracted files are conditionally created and copied into the current directory tree based upon the options described below. The permissions of the files will be those of the previous `cpio -o`. The owner and group of the files will be that of the current user unless the user is super-user, which causes `cpio` to retain the owner and group of the files of the previous `cpio -o`. NOTE: If `cpio -i` tries to create a file that already exists and the existing file is the same age or newer, `cpio` will output a warning message and not replace the file. (The `-u` option can be used to unconditionally overwrite the existing file.)

`cpio -p` (*pass*) reads the standard input to obtain a list of path names of files that are conditionally created and copied into the destination *directory* tree based upon the options described below.

The meanings of the available options are

- `-a` Reset *access* times of input files after they have been copied. Access times are not reset for linked files when `cpio -pla` is specified.
- `-b` Reverse the order of the *bytes* within each word. Use only with the `-i` option.
- `-B` Input/output is to be blocked 5,120 bytes to the record. The default buffer size is 512 bytes when this and the `C` options are not used. (`-B` does not apply to the *pass* option; `-B` is meaningful only with data directed to or from a character special device, e.g. `/dev/rmt/0m`.)
- `-c` Write header information in ASCII *character* form for portability. Always use this option when origin and destination machines are different types.

`-C bufsize`

Input/output is to be blocked *bufsize* bytes to the record, where *bufsize* is replaced by a positive integer. The default buffer size is 512 bytes

- when this and **B** options are not used. (**-C** does not apply to the *pass* option; **-C** is meaningful only with data directed to or from a character special device, e.g. `/dev/rmt/0m`.)
- d** *directories* are to be created as needed.
 - f** Copy in all *files* except those in *patterns*. (See the paragraph on `cpio -i` for a description of *patterns*.)
 - I file** Read the contents of *file* as input. If *file* is a character special device, when the first medium is full replace the medium and type a carriage return to continue to the next medium. Use only with the **-i** option.
 - k** Attempt to skip corrupted file headers and I/O errors that may be encountered. If you want to copy files from a medium that is corrupted or out of sequence, this option lets you read only those files with good headers. (For *cpio* archives that contain other *cpio* archives, if an error is encountered *cpio* may terminate prematurely. *cpio* will find the next good header, which may be one for a smaller archive, and terminate when the smaller archive's trailer is encountered.) Used only with the **-i** option.
 - l** Whenever possible, *link* files rather than copying them. Usable only with the **-p** option.
 - m** Retain previous file *modification* time. This option is ineffective on directories that are being copied.
 - M message**
Define a message to use when switching media. When you use the **-O** or **-I** options and specify a character special device, you can use this option to define the message that is printed when you reach the end of the medium. One `%d` can be placed in the message to print the sequence number of the next medium needed to continue.
 - O file** Direct the output of *cpio* to *file*. If *file* is a character special device, when the first medium is full replace the medium and type a carriage return to continue to the next medium. Use only with the **-o** option.
 - r** Interactively *rename* files. If the user types a null line, the file is skipped. If the user types a "." the original pathname will be copied. (Not available with `cpio -p`.)
 - s** *swap* bytes within each half word. Use only with the **-i** option.
 - S** *Swap* halfwords within each word. Use only with the **-i** option.
 - t** Print a *table of contents* of the input. No files are created.
 - u** Copy *unconditionally* (normally, an older file will not replace a newer file with the same name).
 - v** *verbose*: causes a list of file names to be printed. When used with the **-t** option, the table of contents looks like the output of an `ls -l` command (see `ls (1)`).
 - V** *SpecialVerbose*: print a dot for each file seen. Useful to assure the user that *cpio* is working without printing out all file names.
 - 6** Process an old (i.e. UNIX System *Sixth* Edition format) file. Use only with the **-i** option.

NOTE: *cpio* assumes four-byte words.

If *cpio* reaches end of medium (end of a diskette for example), when writing to (**-o**) or reading from (**-i**) a character special device, and **-O** and **-I** aren't

used, *cpio* will print the message:

If you want to go on, type device/file name when ready.

To continue, you must replace the medium and type the character special device name (*/dev/rdiskette* for example) and carriage return. You may want to continue by directing *cpio* to use a different device. For example, if you have two floppy drives you may want to switch between them so *cpio* can proceed while you are changing the floppies. (A carriage return alone causes the *cpio* process to exit.)

EXAMPLES

The following examples show three uses of *cpio*.

When standard input is directed through a pipe to *cpio -o*, it groups the files so they can be directed (*>*) to a single file (*../newfile*). The *c* option insures that the file will be portable to other machines. Instead of *ls(1)*, you could use *find(1)*, *echo(1)*, *cat(1)*, etc. to pipe a list of names to *cpio*. You could direct the output to a device instead of a file.

```
ls | cpio -oc > ../newfile
```

cpio -i uses the output file of *cpio -o* (directed through a pipe with *cat* in the example), extracts those files that match the patterns (*memo/a1*, *memo/b**), creates directories below the current directory as needed (*-d* option), and places the files in the appropriate directories. The *c* option is used when the file is created with a portable header. If no patterns were given, all files from *newfile* would be placed in the directory.

```
cat newfile | cpio -icd "memo/a1" "memo/b"
```

cpio -p takes the file names piped to it and copies or links (*-l* option) those files to another directory on your machine (*newdir* in the example). The *-d* options says to create directories as needed. The *-m* option says retain the modification time. (It is important to use the *-depth* option of *find(1)* to generate path names for *cpio*. This eliminates problems *cpio* could have trying to create files under read-only directories.)

```
find . -depth -print | cpio -pdlmv newdir
```

SEE ALSO

ar(1), *cat(1)*, *echo(1)*, *find(1)*, *ls(1)*, *tar(1)*.
cpio(4) in the *System Administrator's Reference Manual*.

NOTES

- 1) Path names are restricted to 256 characters.
- 2) Only the super-user can copy special files.
- 3) Blocks are reported in 512-byte quantities.
- 4) If a file has 000 permissions, contains more than 0 characters of data, and the user is not root, the file will not be saved or restored.



NAME

`crontab` – user crontab file

SYNOPSIS

```
crontab [file]
crontab -r
crontab -l
```

DESCRIPTION

`crontab` copies the specified file, or standard input if no file is specified, into a directory that holds all users' crontabs. The `-r` option removes a user's crontab from the crontab directory. `crontab -l` will list the crontab file for the invoking user.

Users are permitted to use `crontab` if their names appear in the file `/usr/lib/cron/cron.allow`. If that file does not exist, the file `/usr/lib/cron/cron.deny` is checked to determine if the user should be denied access to `crontab`. If neither file exists, only root is allowed to submit a job. If `cron.allow` does not exist and `cron.deny` exists but is empty, global usage is permitted. The allow/deny files consist of one user name per line.

A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns that specify the following:

```
minute (0–59),
hour (0–23),
day of the month (1–31),
month of the year (1–12),
day of the week (0–6 with 0=Sunday).
```

Each of these patterns may be either an asterisk (meaning all legal values) or a list of elements separated by commas. An element is either a number or two numbers separated by a minus sign (meaning an inclusive range). Note that the specification of days may be made by two fields (day of the month and day of the week). If both are specified as a list of elements, both are adhered to. For example, `0 0 1,15 * 1` would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to `*` (for example, `0 0 * * 1` would run a command only on Mondays).

The sixth field of a line in a crontab file is a string that is executed by the shell at the specified times. A percent character in this field (unless escaped by `\`) is translated to a new-line character. Only the first line (up to a `%` or end of line) of the command field is executed by the shell. The other lines are made available to the command as standard input.

The shell is invoked from your `$HOME` directory with an `arg0` of `sh`. Users who desire to have their *profile* executed must explicitly do so in the crontab file. `Cron` supplies a default environment for every shell, defining `HOME`, `LOGNAME`, `SHELL(=/bin/sh)`, and `PATH(=/bin:/usr/bin:/usr/sbin)`.

If you do not redirect the standard output and standard error of your commands, any generated output or errors will be mailed to you.

FILES

/usr/lib/cron	main cron directory
/usr/spool/cron/crontabs	spool area
/usr/lib/cron/log	accounting information
/usr/lib/cron/cron.allow	list of allowed users
/usr/lib/cron/cron.deny	list of denied users

SEE ALSO

sh(1).
cron(1M) in the *System Administrator's Reference Manual*.

WARNINGS

If you inadvertently enter the **crontab** command with no argument(s), do not attempt to get out with a CTRL-d. This will cause all entries in your **crontab** file to be removed. Instead, exit with a DEL.

NAME

`crypt` – encode/decode

SYNOPSIS

```
crypt [ password ]
crypt [-k]
```

DESCRIPTION

`crypt` reads from the standard input and writes on the standard output. The *password* is a key that selects a particular transformation. If no argument is given, `crypt` demands a key from the terminal and turns off printing while the key is being typed in. If the `-k` option is used, `crypt` will use the key assigned to the environment variable `CRYPTKEY`. `crypt` encrypts and decrypts with the same key:

```
crypt key <clear >cypher
crypt key <cypher | pr
```

Files encrypted by `crypt` are compatible with those treated by the editors `ed(1)`, `edit(1)`, `ex(1)`, and `vi(1)` in encryption mode.

The security of encrypted files depends on three factors: the fundamental method must be hard to solve; direct search of the key space must be infeasible; “sneak paths” by which keys or clear text can become visible must be minimized.

`crypt` implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are known, but not widely; moreover the amount of work required is likely to be large.

The transformation of a key into the internal settings of the machine is deliberately designed to be expensive, i.e., to take a substantial fraction of a second to compute. However, if keys are restricted to (say) three lower-case letters, then encrypted files can be read by expending only a substantial fraction of five minutes of machine time.

If the key is an argument to the `crypt` command, it is potentially visible to users executing `ps(1)` or a derivative. The choice of keys and key security are the most vulnerable aspect of `crypt`.

FILES

`/dev/tty` for typed key

SEE ALSO

`ed(1)`, `edit(1)`, `ex(1)`, `makekey(1)`, `ps(1)`, `stty(1)`, `vi(1)`.

WARNING

This command is provided with the Security Administration Utilities, which is only available in the United States. If two or more files encrypted with the same key are concatenated and an attempt is made to decrypt the result, only the contents of the first of the original files will be decrypted correctly.

BUGS

If output is piped to `nroff` and the encryption key is *not* given on the command line, `crypt` can leave terminal modes in a strange state (see `stty(1)`).



NAME

`csplit` – context split

SYNOPSIS

`csplit` [-s] [-k] [-f prefix] file arg1 [... argn]

DESCRIPTION

`csplit` reads *file* and separates it into *n*+1 sections, defined by the arguments *arg1*... *argn*. By default the sections are placed in *xx00* ... *xxn* (*n* may not be greater than 99). These sections get the following pieces of *file*:

- 00: From the start of *file* up to (but not including) the line referenced by *arg1*.
- 01: From the line referenced by *arg1* up to the line referenced by *arg2*.
- ⋮
- n*+1: From the line referenced by *argn* to the end of *file*.

If the *file* argument is a `-` then standard input is used.

The options to `csplit` are:

- `-s` `csplit` normally prints the character counts for each file created. If the `-s` option is present, `csplit` suppresses the printing of all character counts.
- `-k` `csplit` normally removes created files if an error occurs. If the `-k` option is present, `csplit` leaves previously created files intact.
- `-f prefix` If the `-f` option is used, the created files are named *prefix00* ... *prefixn*. The default is *xx00* ... *xxn*.

The arguments (*arg1* ... *argn*) to `csplit` can be a combination of the following:

- /rexp/* A file is to be created for the section from the current line up to (but not including) the line containing the regular expression *rexp*. The current line becomes the line containing *rexp*. This argument may be followed by an optional `+` or `-` some number of lines (e.g., */Page/-5*).
- %rexp%* This argument is the same as */rexp/*, except that no file is created for the section.
- lnno* A file is to be created from the current line up to (but not including) *lnno*. The current line becomes *lnno*.
- {num}* Repeat argument. This argument may follow any of the above arguments. If it follows a *rexp* type argument, that argument is applied *num* more times. If it follows *lnno*, the file will be split every *lnno* lines (*num* times) from that point.

Enclose all *rexp* type arguments that contain blanks or other characters meaningful to the shell in the appropriate quotes. Regular expressions may not contain embedded new-lines. `csplit` does not affect the original file; it is the users responsibility to remove it.

EXAMPLES

```
csplit -f cobol file '/procedure division/' /par5./ /par16./
```

This example creates four files, **cobol00** ... **cobol03**. After editing the "split" files, they can be recombined as follows:

```
cat cobol0[0-3] > file
```

Note that this example overwrites the original file.

```
csplit -k file 100 {99}
```

This example would split the file at every 100 lines, up to 10,000 lines. The **-k** option causes the created files to be retained if there are less than 10,000 lines; however, an error message would still be printed.

```
csplit -k prog.c '%main(%' '/'^)/+1' {20}
```

Assuming that **prog.c** follows the normal C coding convention of ending routines with a **}** at the beginning of the line, this example will create a file containing each separate C routine (up to 21) in **prog.c**.

SEE ALSO

ed(1), **sh(1)**.

regex(5) in the *System Administrator's Reference Manual*.

DIAGNOSTICS

Self-explanatory except for:

arg - out of range

which means that the given argument did not reference a line between the current position and the end of the file.

NAME

`ct` - spawn `getty` to a remote terminal

SYNOPSIS

`ct [-wn] [-xn] [-h] [-v] [-sspeed] telno ...`

DESCRIPTION

`ct` dials the telephone number of a modem that is attached to a terminal, and spawns a `getty` process to that terminal. `Telno` is a telephone number, with equal signs for secondary dial tones and minus signs for delays at appropriate places. (The set of legal characters for `telno` is 0 thru 9, -, =, *, and #. The maximum length `telno` is 31 characters). If more than one telephone number is specified, `ct` will try each in succession until one answers; this is useful for specifying alternate dialing paths.

`ct` will try each line listed in the file `/usr/lib/uucp/Devices` until it finds an available line with appropriate attributes or runs out of entries. If there are no free lines, `ct` will ask if it should wait for one, and if so, for how many minutes it should wait before it gives up. `ct` will continue to try to open the dialers at one-minute intervals until the specified limit is exceeded. The dialogue may be overridden by specifying the `-wn` option, where `n` is the maximum number of minutes that `ct` is to wait for a line.

The `-xn` option is used for debugging; it produces a detailed output of the program execution on `stderr`. The debugging level, `n`, is a single digit; `-x9` is the most useful value.

Normally, `ct` will hang up the current line, so the line can answer the incoming call. The `-h` option will prevent this action. The `-h` option will also wait for the termination of the specified `ct` process before returning control to the user's terminal. If the `-v` option is used, `ct` will send a running narrative to the standard error output stream.

The data rate may be set with the `-s` option, where `speed` is expressed in baud. The default rate is 1200.

After the user on the destination terminal logs out, there are two things that could occur depending on what type of `getty` is on the line (`getty` or `uugetty`). For the first case, `ct` prompts, **Reconnect?** If the response begins with the letter `n`, the line will be dropped; otherwise, `getty` will be started again and the **login:** prompt will be printed. In the second case, there is already a `getty` (`uugetty`) on the line, so the **login:** message will appear.

To log out properly, the user must type **control D**.

Of course, the destination terminal must be attached to a modem that can answer the telephone.

FILES

`/usr/lib/uucp/Devices`
`/usr/adm/ctlog`

SEE ALSO

`cu(1C)`, `login(1)`, `uucp(1C)`,
`getty(1M)`, `uugetty(1M)` in the *System Administrator's Reference Manual*.

BUGS

For a shared port, one used for both dial-in and dial-out, the *uugetty* program running on the line must have the `-r` option specified (see *uugetty(1M)*).

NAME

cu – call another UNIX system

SYNOPSIS

```
cu [-sspeed] [-lline] [-h] [-t] [-d] [-o | -e] [-n] telno
cu [ -s speed ] [ -h ] [ -d ] [ -o | -e ] -l line
cu [-h] [-d] [-o | -e] systemname
```

DESCRIPTION

cu calls up another UNIX system, a terminal, or possibly a non-UNIX system. It manages an interactive conversation with possible transfers of ASCII files.

cu accepts the following options and arguments:

- sspeed** Specifies the transmission speed (300, 1200, 2400, 4800, 9600); The default value is "Any" speed which will depend on the order of the lines in the `/usr/lib/uucp/Devices` file. Most modems are either 300 or 1200 baud. Directly connected lines may be set to a speed higher than 1200 baud.
- lline** Specifies a device name to use as the communication line. This can be used to override the search that would otherwise take place for the first available line having the right speed. When the **-l** option is used without the **-s** option, the speed of a line is taken from the `Devices` file. When the **-l** and **-s** options are both used together, **cu** will search the `Devices` file to check if the requested speed for the requested line is available. If so, the connection will be made at the requested speed; otherwise an error message will be printed and the call will not be made. The specified device is generally a directly connected asynchronous line (e.g., `/dev/ttyab`) in which case a telephone number (*telno*) is not required. The specified device need not be in the `/dev` directory. If the specified device is associated with an auto dialer, a telephone number must be provided. Use of this option with *systemname* rather than *telno* will not give the desired result (see *systemname* below).
- h** Emulates local echo, supporting calls to other computer systems which expect terminals to be set to half-duplex mode.
- t** Used to dial an ASCII terminal which has been set to auto answer. Appropriate mapping of carriage-return to carriage-return-line-feed pairs is set.
- d** Causes diagnostic traces to be printed.
- o** Designates that odd parity is to be generated for data sent to the remote system.
- n** For added security, will prompt the user to provide the telephone number to be dialed rather than taking it from the command line.
- e** Designates that even parity is to be generated for data sent to the remote system.

- telno* When using an automatic dialer, the argument is the telephone number with equal signs for secondary dial tone or minus signs placed appropriately for delays of 4 seconds.
- systemname* A uucp system name may be used rather than a telephone number; in this case, *cu* will obtain an appropriate direct line or telephone number from `/usr/lib/uucp/Systems`. Note: the *systemname* option should not be used in conjunction with the `-l` and `-s` options as *cu* will connect to the first available line for the system name specified, ignoring the requested line and speed.

After making the connection, *cu* runs as two processes: the *transmit* process reads data from the standard input and, except for lines beginning with `~`, passes it to the remote system; the *receive* process accepts data from the remote system and, except for lines beginning with `~`, passes it to the standard output. Normally, an automatic DC3/DC1 protocol is used to control input from the remote so the buffer is not overrun. Lines beginning with `~` have special meanings.

The *transmit* process interprets the following user initiated commands:

- `~.` terminate the conversation.
- `~!` escape to an interactive shell on the local system.
- `~!cmd...` run *cmd* on the local system (via `sh -c`).
- `~$cmd...` run *cmd* locally and send its output to the remote system.
- `~%cd` change the directory on the local system. Note: `~!cd` will cause the command to be run by a sub-shell, probably not what was intended.
- `~%take from [to]` copy file *from* (on the remote system) to file *to* on the local system. If *to* is omitted, the *from* argument is used in both places.
- `~%put from [to]` copy file *from* (on local system) to file *to* on remote system. If *to* is omitted, the *from* argument is used in both places.

For both `~%take` and `put` commands, as each block of the file is transferred, consecutive single digits are printed to the terminal.

- `~ line` send the line *line* to the remote system.
- `~%break` transmit a **BREAK** to the remote system (which can also be specified as `~%b`).
- `~%debug` toggles the `-d` debugging option on or off (which can also be specified as `~%d`).
- `~t` prints the values of the `termio` structure variables for the user's terminal (useful for debugging).

- `~l` prints the values of the termio structure variables for the remote communication line (useful for debugging).
- `~%nostop` toggles between DC3/DC1 input control protocol and no input control. This is useful in case the remote system is one which does not respond properly to the DC3 and DC1 characters.

The *receive* process normally copies data from the remote system to its standard output. Internally the program accomplishes this by initiating an output diversion to a file when a line from the remote begins with `~`.

Data from the remote is diverted (or appended, if `>>` is used) to *file* on the local system. The trailing `~>` marks the end of the diversion.

The use of `~%put` requires *stty(1)* and *cat(1)* on the remote side. It also requires that the current erase and kill characters on the remote system be identical to these current control characters on the local system. Backslashes are inserted at appropriate places.

The use of `~%take` requires the existence of *echo(1)* and *cat(1)* on the remote system. Also, *tabs* mode (See *stty(1)*) should be set on the remote system if tabs are to be copied without expansion to spaces.

When *cu* is used on system X to connect to system Y and subsequently used on system Y to connect to system Z, commands on system Y can be executed by using `~.` Executing a tilde command reminds the user of the local system *uname*. For example, *uname* can be executed on Z, X, and Y as follows:

```
uname
Z
[X]!uname
X
~[Y]!uname
Y
```

In general, `~` causes the command to be executed on the original machine, `~.` causes the command to be executed on the next machine in the chain.

EXAMPLES

To dial a system whose telephone number is 9 201 555 1212 using 1200 baud (where dialtone is expected after the 9):

```
cu -s1200 9-12015551212
```

If the speed is not specified, "Any" is the default value.

To login to a system connected by a direct line:

```
cu -l /dev/ttyXX
```

or

```
cu -l ttyXX
```

To dial a system with the specific line and a specific speed:

```
cu -s1200 -l ttyXX
```

To dial a system using a specific line associated with an auto dialer:
`cu -l cuLXX 9=12015551212`

To use a system name:
`cu systemname`

FILES

`/usr/lib/uucp/Systems`
`/usr/lib/uucp/Devices`
`/usr/spool/locks/LCK..(tty-device)`

SEE ALSO

`cat(1)`, `ct(1C)`, `echo(1)`, `stty(1)`, `uucp(1C)`, `uname(1)`.

DIAGNOSTICS

Exit code is zero for normal exit, otherwise, one.

WARNINGS

The `cu` command does not do any integrity checking on data it transfers. Data fields with special `cu` characters may not be transmitted properly. Depending on the interconnection hardware, it may be necessary to use a `~` to terminate the conversion even if `stty 0` has been used. Non-printing characters are not dependably transmitted using either the `~%put` or `~%take` commands. `cu` between an IMBR1 and a penril modem will not return a login prompt immediately upon connection. A carriage return will return the prompt.

BUGS

There is an artificial slowing of transmission by `cu` during the `~%put` operation so that loss of data is unlikely.

NAME

`cut` - cut out selected fields of each line of a file

SYNOPSIS

```
cut -c list [file ...]
cut -f list [-d char] [-s] [file ...]
```

DESCRIPTION

Use `cut` to cut out columns from a table or fields from each line of a file; in data base parlance, it implements the projection of a relation. The fields as specified by *list* can be fixed length, i.e., character positions as on a punched card (`-c` option) or the length can vary from line to line and be marked with a field delimiter character like *tab* (`-f` option). `cut` can be used as a filter; if no files are given, the standard input is used. In addition, a file name of "-" explicitly refers to standard input.

The meanings of the options are:

- list* A comma-separated list of integer field numbers (in increasing order), with optional `-` to indicate ranges [e.g., 1,4,7; 1-3,8; -5,10 (short for 1-5,10); or 3- (short for third through last field)].
- `-c list` The *list* following `-c` (no space) specifies character positions (e.g., `-c1-72` would pass the first 72 characters of each line).
- `-f list` The *list* following `-f` is a list of fields assumed to be separated in the file by a delimiter character (see `-d`); e.g., `-f1,7` copies the first and seventh field only. Lines with no field delimiters will be passed through intact (useful for table subheadings), unless `-s` is specified.
- `-d char` The character following `-d` is the field delimiter (`-f` option only). Default is *tab*. Space or other characters with special meaning to the shell must be quoted.
- `-s` Suppresses lines with no delimiter characters in case of `-f` option. Unless specified, lines with no delimiters will be passed through untouched.

Either the `-c` or `-f` option must be specified.

Use `grep(1)` to make horizontal "cuts" (by context) through a file, or `paste(1)` to put files together column-wise (i.e., horizontally). To reorder columns in a table, use `cut` and `paste`.

EXAMPLES

```
cut -d: -f1,5 /etc/passwd           mapping of user IDs to names
name=`who am i |cut -f1 -d" "`     to set name to current login name.
```

DIAGNOSTICS

ERROR: line too long A line can have no more than 1023 characters or fields, or there is no new-line character.

ERROR: bad list for c/f option Missing `-c` or `-f` option or incorrectly specified *list*. No error occurs if a line has fewer fields than the *list* calls for.

ERROR: no fields The list is empty.

ERROR: no delimiter Missing *char* on **-d** option.

ERROR: cannot handle multiple adjacent backspaces
Adjacent backspaces cannot be processed correctly.

WARNING: cannot open <filename>
Either *filename* cannot be read or does not exist. If multiple filenames are present, processing continues.

SEE ALSO

grep(1), paste(1).

NAME

date – print and set the date

SYNOPSIS

date [+format]
 date [mmddhhmm[[yy] | [ccyy]]]

DESCRIPTION

If no argument is given, or if the argument begins with +, the current date and time are printed. Otherwise, the current date is set (only by super-user). The first *mm* is the month number; *dd* is the day number in the month; *hh* is the hour number (24 hour system); the second *mm* is the minute number; *cc* is the century minus one and is optional; *yy* is the last 2 digits of the year number and is optional. For example:

date 10080045

sets the date to Oct 8, 12:45 AM. The current year is the default if no year is mentioned. The system operates in GMT. *date* takes care of the conversion to and from local standard and daylight time. Only the super-user may change the date.

If the argument begins with +, the output of *date* is under the control of the user. All output fields are of fixed size (zero padded if necessary). Each Field Descriptor is preceded by % and will be replaced in the output by its corresponding value. A single % is encoded by %%. All other characters are copied to the output without change. The string is always terminated with a new-line character. If the argument contains embedded blanks it must be quoted (see the EXAMPLE section).

Specifications of native language translations of month and weekday names are supported. The language used depends on the value of the environment variable LANGUAGE (see *environ*(5)). The month and weekday names used for a language are taken from strings in the file for that language in the */lib/cftime* directory (see *cftime*(4)).

After successfully setting the date and time, *date* will display the new date according to the format defined in the environment variable CFTIME (see *environ*(5)).

Field Descriptors (must be preceded by a %):

a	abbreviated weekday name
A	full weekday name
b	abbreviated month name
B	full month name
d	day of month – 01 to 31
D	date as mm/dd/yy
e	day of month – 1 to 31 (single digits are preceded by a blank)
h	abbreviated month name (alias for %b)
H	hour – 00 to 23
I	hour – 01 to 12 ^h
j	day of year – 001 to 366
m	month of year – 01 to 12

M	minute – 00 to 59
n	insert a new-line character
p	string containing ante-meridiem or post-meridiem indicator (by default, AM or PM)
r	time as <i>hh:mm:ss pp</i> where <i>pp</i> is the ante-meridiem or post-meridiem indicator (by default, AM or PM)
R	time as <i>hh:mm</i>
S	second – 00 to 59
t	insert a tab character
T	time as <i>hh:mm:ss</i>
U	week number of year (Sunday as the first day of the week) – 01 to 52
w	day of week – Sunday = 0
W	week number of year (Monday as the first day of the week) – 01 to 52
x	Country-specific date format
X	Country-specific time format
y	year within century – 00 to 99
Y	year as <i>ccyy</i> (4 digits)
Z	timezone name

EXAMPLE

date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'

would have generated as output:

DATE: 08/01/76
TIME: 14:45:05

DIAGNOSTICS

No permission if you are not the super-user and you try to change the date
bad conversion if the date set is syntactically incorrect

FILES

/dev/kmem

WARNING

Should you need to change the date while the system is running multi-user, use *sysadm(1) datetime*.

NOTE

Administrators should note the following: if you attempt to set the current date to one of the dates that the standard and alternate time zones change (for example, the date that daylight time is starting or ending), and you attempt to set the time to a time in the interval between the end of standard time and the beginning of the alternate time (or the end of the alternate time and the beginning of standard time), the results are unpredictable.

SEE ALSO

sysadm(1).
cftime(4), *environ(5)* in the *System Administrator's Reference Manual*.

NAME

dc – desk calculator

SYNOPSIS

dc [file]

DESCRIPTION

dc is an arbitrary precision arithmetic package. Ordinarily it operates on decimal integers, but one may specify an input base, output base, and a number of fractional digits to be maintained. (See *bc(1)*, a preprocessor for *dc* that provides infix notation and a C-like syntax that implements functions. *Bc* also provides reasonable control structures for programs.) The overall structure of *dc* is a stacking (reverse Polish) calculator. If an argument is given, input is taken from that file until its end, then from the standard input. The following constructions are recognized:

number

The value of the number is pushed on the stack. A number is an unbroken string of the digits 0–9. It may be preceded by an underscore (`_`) to input a negative number. Numbers may contain decimal points.

`+ - / * % ^`

The top two values on the stack are added (+), subtracted (-), multiplied (*), divided (/), remaindered (%), or exponentiated (^). The two entries are popped off the stack; the result is pushed on the stack in their place. Any fractional part of an exponent is ignored.

`sx` The top of the stack is popped and stored into a register named *x*, where *x* may be any character. If the *s* is capitalized, *x* is treated as a stack and the value is pushed on it.

`lx` The value in register *x* is pushed on the stack. The register *x* is not altered. All registers start with zero value. If the *l* is capitalized, register *x* is treated as a stack and its top value is popped onto the main stack.

`d` The top value on the stack is duplicated.

`p` The top value on the stack is printed. The top value remains unchanged.

`P` Interprets the top of the stack as an ASCII string, removes it, and prints it.

`f` All values on the stack are printed.

`q` Exits the program. If executing a string, the recursion level is popped by two.

`Q` Exits the program. The top value on the stack is popped and the string execution level is popped by that value.

`x` Treats the top element of the stack as a character string and executes it as a string of *dc* commands.

`X` Replaces the number on the top of the stack with its scale factor.

- [...] Puts the bracketed ASCII string onto the top of the stack.
- <x >x =x
The top two elements of the stack are popped and compared. Register *x* is evaluated if they obey the stated relation.
- v Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.
- ! Interprets the rest of the line as a UNIX system command.
- c All values on the stack are popped.
- i The top value on the stack is popped and used as the number radix for further input. I Pushes the input base on the top of the stack.
- o The top value on the stack is popped and used as the number radix for further output.
- O Pushes the output base on the top of the stack.
- k The top of the stack is popped, and that value is used as a non-negative scale factor: the appropriate number of places are printed on output, and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.
- z The stack level is pushed onto the stack.
- Z Replaces the number on the top of the stack with its length.
- ? A line of input is taken from the input source (usually the terminal) and executed.
- ;: are used by *bc(1)* for array operations.

EXAMPLE

This example prints the first ten values of *n!*:

```
[la1+dsa*pla10>y]sy
0sa1
lyx
```

SEE ALSO

bc(1).

DIAGNOSTICS

x is unimplemented
where *x* is an octal number.

stack empty
for not enough elements on the stack to do what was asked.

Out of space
when the free list is exhausted (too many digits).

Out of headers
for too many numbers being kept around.

Out of pushdown
for too many items on the stack.

Nesting Depth
for too many levels of nested execution.





NAME

dd – convert and copy a file

SYNOPSIS

dd [option=value] ...

DESCRIPTION

dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block size may be specified to take advantage of raw physical I/O.

<i>option</i>	<i>values</i>
if=file	input file name; standard input is default
of=file	output file name; standard output is default
ibs=n	input block size <i>n</i> bytes (default 512)
obs=n	output block size (default 512)
bs=n	set both input and output block size, superseding <i>ibs</i> and <i>obs</i> ; also, if no conversion is specified, it is particularly efficient since no in-core copy need be done
cbs=n	conversion buffer size
skip=n	skip <i>n</i> input blocks before starting copy
seek=n	seek <i>n</i> blocks from beginning of output file before copying
count=n	copy only <i>n</i> input blocks
conv=ascii	convert EBCDIC to ASCII
ebcdic	convert ASCII to EBCDIC
ibm	slightly different map of ASCII to EBCDIC
lcase	map alphabetic to lower case
ucase	map alphabetic to upper case
swab	swap every pair of bytes
noerror	do not stop processing on an error
sync	pad every input block to <i>ibs</i>
... , ...	several comma-separated conversions

Where sizes are specified, a number of bytes is expected. A number may end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by **x** to indicate multiplication.

cbs is used only if *conv=ascii* or *conv=ebcdic* is specified. In the former case, *cbs* characters are placed into the conversion buffer (converted to ASCII). Trailing blanks are trimmed and a new-line added before sending the line to the output. In the latter case, ASCII characters are read into the conversion buffer (converted to EBCDIC). Blanks are added to make up an output block of size *cbs*.

After completion, *dd* reports the number of whole and partial input and output blocks.

DIAGNOSTICS

f+p blocks in(out) numbers of full and partial blocks read(written)



NAME

deroff – remove *nroff*/*troff*, *tbl*, and *eqn* constructs

SYNOPSIS

deroff [**-mx**] [**-w**] [files]

DESCRIPTION

deroff reads each of the *files* in sequence and removes all *troff*(1) requests, macro calls, backslash constructs, *eqn*(1) constructs (between *.EQ* and *.EN* lines, and between delimiters), and *tbl*(1) descriptions, perhaps replacing them with white space (blanks and blank lines), and writes the remainder of the file on the standard output. *deroff* follows chains of included files (*.so* and *.nx troff* commands); if a file has already been included, a *.so* naming that file is ignored and a *.nx* naming that file terminates execution. If no input file is given, *deroff* reads the standard input.

The **-m** option may be followed by an **m**, **s**, or **l**. The **-mm** option causes the macros to be interpreted so that only running text is output (i.e., no text from macro lines.) The **-ml** option forces the **-mm** option and also causes deletion of lists associated with the **mm** macros.

If the **-w** option is given, the output is a word list, one "word" per line, with all other characters deleted. Otherwise, the output follows the original, with the deletions mentioned above. In text, a "word" is any string that *contains* at least two letters and is composed of letters, digits, ampersands (&), and apostrophes ('); in a macro call, however, a "word" is a string that *begins* with at least two letters and contains a total of at least three letters. Delimiters are any characters other than letters, digits, apostrophes, and ampersands. Trailing apostrophes and ampersands are removed from "words."

SEE ALSO

eqn(1), *nroff*(1), *tbl*(1), *troff*(1) in the *DOCUMENTER'S WORKBENCH Software Release 2.0 Technical Discussion and Reference Manual*.

BUGS

deroff is not a complete *troff* interpreter, so it can be confused by subtle constructs. Most such errors result in too much rather than too little output. The **-ml** option does not handle nested lists correctly.



NAME

df – report number of free disk blocks and i-nodes

SYNOPSIS

df [**-lt**] [**-f**] [*file-system* | *directory* | *mounted-resource*]

DESCRIPTION

The **df** command prints out the number of free blocks and free i-nodes in mounted file systems, directories, or mounted resources by examining the counts kept in the super-blocks.

file-system may be specified either by device name (e.g., **/dev/dsk/c1d0s2**) or by mount point directory name (e.g., **/usr**).

directory can be a directory name. The report presents information for the device that contains the directory.

mounted-resource can be a remote resource name. The report presents information for the remote device that contains the resource.

If no arguments are used, the free space on all locally and remotely mounted file systems is printed.

The **df** command uses the following options:

- l** only reports on local file systems.
- t** causes the figures for total allocated blocks and i-nodes to be reported as well as the free blocks and i-nodes.
- f** an actual count of the blocks in the free list is made, rather than taking the figure from the super-block (free i-nodes are not reported). This option will not print any information about mounted remote resources.

NOTE

If multiple remote resources are listed that reside on the same file system on a remote machine, each listing after the first one will be marked with an asterisk.

FILES

/dev/dsk/*
/etc/mnttab

SEE ALSO

mount(1M), **fs(4)**, **mnttab(4)** in the *System Administrator's Reference Manual*.



NAME

diff – differential file comparator

SYNOPSIS

diff [**-efbh**] file1 file2

DESCRIPTION

diff tells what lines must be changed in two files to bring them into agreement. If *file1* (*file2*) is **-**, the standard input is used. If *file1* (*file2*) is a directory, then a file in that directory with the name *file2* (*file1*) is used. The normal output contains lines of these forms:

```
n1 a n3,n4
n1,n2 d n3
n1,n2 c n3,n4
```

These lines resemble *ed* commands to convert *file1* into *file2*. The numbers after the letters pertain to *file2*. In fact, by exchanging **a** for **d** and reading backward one may ascertain equally how to convert *file2* into *file1*. As in *ed*, identical pairs, where $n1 = n2$ or $n3 = n4$, are abbreviated as a single number.

Following each of these lines come all the lines that are affected in the first file flagged by **<**, then all the lines that are affected in the second file flagged by **>**.

The **-b** option causes trailing blanks (spaces and tabs) to be ignored and other strings of blanks to compare equal.

The **-e** option produces a script of *a*, *c*, and *d* commands for the editor *ed*, which will recreate *file2* from *file1*. The **-f** option produces a similar script, not useful with *ed*, in the opposite order. In connection with **-e**, the following shell program may help maintain multiple versions of a file. Only an ancestral file (\$1) and a chain of version-to-version *ed* scripts (\$2,\$3,...) made by *diff* need be on hand. A "latest version" appears on the standard output.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Except in rare circumstances, *diff* finds a smallest sufficient set of file differences.

Option **-h** does a fast, half-hearted job. It works only when changed stretches are short and well separated, but does work on files of unlimited length. Options **-e** and **-f** are unavailable with **-h**.

FILES

```
/tmp/d?????
/usr/lib/diffh for -h
```

SEE ALSO

bdiff(1), cmp(1), comm(1), ed(1).

DIAGNOSTICS

Exit status is 0 for no differences, 1 for some differences, 2 for trouble.

BUGS

Editing scripts produced under the **-e** or **-f** option are naive about creating lines consisting of a single period (.).

WARNINGS

Missing newline at end of file X
indicates that the last line of file X did not have a new-line. If the lines are different, they will be flagged and output; although the output will seem to indicate they are the same.

NAME

`diff3` - 3-way differential file comparison

SYNOPSIS

`diff3 [-ex3] file1 file2 file3`

DESCRIPTION

`diff3` compares three versions of a file, and publishes disagreeing ranges of text flagged with these codes:

```

-----      all three files differ
-----1     file1 is different
-----2     file2 is different
-----3     file3 is different

```

The type of change suffered in converting a given range of a given file to some other is indicated in one of these ways:

```

f : n1 a      Text is to be appended after line number n1 in file f,
               where f = 1, 2, or 3.
f : n1 , n2 c  Text is to be changed in the range line n1 to line n2.
               If n1 = n2, the range may be abbreviated to n1.

```

The original contents of the range follows immediately after a `c` indication. When the contents of two files are identical, the contents of the lower-numbered file is suppressed.

Under the `-e` option, `diff3` publishes a script for the editor `ed` that will incorporate into `file1` all changes between `file2` and `file3`, i.e., the changes that normally would be flagged `-----` and `-----3`. Option `-x` (`-3`) produces a script to incorporate only changes flagged `-----` (`-----3`). The following command will apply the resulting script to `file1`.

```
(cat script; echo '1,$p') | ed - file1
```

FILES

```

/tmp/d3*
/usr/lib/diff3prog

```

SEE ALSO

`diff(1)`.

BUGS

Text lines that consist of a single `.` will defeat `-e`.
Files longer than 64K bytes will not work.



NAME

dircmp – directory comparison

SYNOPSIS

dircmp [**-d**] [**-s**] [**-wn**] dir1 dir2

DESCRIPTION

dircmp examines *dir1* and *dir2* and generates various tabulated information about the contents of the directories. Listings of files that are unique to each directory are generated for all the options. If no option is entered, a list is output indicating whether the file names common to both directories have the same contents.

- d** Compare the contents of files with the same name in both directories and output a list telling what must be changed in the two files to bring them into agreement. The list format is described in *diff*(1).
- s** Suppress messages about identical files.
- wn** Change the width of the output line to *n* characters. The default width is 72.

SEE ALSO

cmp(1), *diff*(1).



NAME

`dsconfig` – display data storage device configuration

SYNOPSIS

`/usr/bin/dsconfig` [*simple_administration_device_name*]

DESCRIPTION

The `dsconfig` command produces the mapping of the simple administration names for data storage devices found in `/dev/rSA` to the device names found in `/dev/rdisk` or `/dev/rmt` and prints the physical location of the associated peripheral on the machine. The `dsconfig` command with no arguments prints the mapping for every entry in `/dev/rSA`.

EXAMPLE

```
dsconfig disk1 disk6
```

```
SA: disk1
```

```
device: /dev/rdisk/c1d0s6
```

```
configuration: Integral Disk Drive 0
```

```
SA: disk6
```

```
device: /dev/rdisk/c1t5d2s6
```

```
configuration: Slot 1 Target Controller 5 Drive 2
```



NAME

`du` – summarize disk usage

SYNOPSIS

`du [-sar] [names]`

DESCRIPTION

du reports the number of blocks contained in all files and (recursively) directories within each directory and file specified by the *names* argument. The block count includes the indirect blocks of the file. If *names* is missing, the current directory is used.

The optional arguments are as follows:

- `-s` causes only the grand total (for each of the specified *names*) to be given.
- `-a` causes an output line to be generated for each file.

If neither `-s` or `-a` is specified, an output line is generated for each directory only.

- `-r` will cause *du* to generate messages about directories that cannot be read, files that cannot be opened, etc., rather than being silent (the default).

A file with two or more links is only counted once.

BUGS

If the `-a` option is not used, non-directories given as arguments are not listed. If there are links between files in different directories where the directories are on separate branches of the file system hierarchy, *du* will count the excess files more than once.

Files with holes in them will get an incorrect block count. (See Chapter 5, File System Administration, in the *System Administrator's Guide*)



NAME

echo – echo arguments

SYNOPSIS

echo [arg] ...

DESCRIPTION

echo writes its arguments separated by blanks and terminated by a new-line on the standard output. It also understands C-like escape conventions; beware of conflicts with the shell's use of \:

\b	backspace
\c	print line without new-line
\f	form-feed
\n	new-line
\r	carriage return
\t	tab
\v	vertical tab
\\	backslash
\0n	where <i>n</i> is the 8-bit character whose ASCII code is the 1-, 2- or 3-digit octal number representing that character.

echo is useful for producing diagnostics in command files and for sending known data into a pipe.

SEE ALSO

sh(1).

CAVEATS

When representing an 8-bit character by using the escape convention `\0n`, the *n* must **always** be preceded by the digit zero (0).

For example, typing: `echo 'WARNING:\07'` will print the phrase **WARNING:** and sound the "bell" on your terminal. The use of single (or double) quotes (or two backslashes) is required to protect the "\" that precedes the "07".

For the octal equivalents of each character, see `ascii(5)`, in the *System Administrator's Reference Manual*.



NAME

`ed`, `red` – text editor

SYNOPSIS

`ed` [`-s`] [`-p` string] [`-x`] [`-C`] [file]

`red` [`-s`] [`-p` string] [`-x`] [`-C`] [file]

DESCRIPTION

`ed` is the standard text editor. If the *file* argument is given, `ed` simulates an *e* command (see below) on the named file; that is to say, the file is read into `ed`'s buffer so that it can be edited.

- `-s` Suppresses the printing of character counts by *e*, *r*, and *w* commands, of diagnostics from *e* and *q* commands, and of the *!* prompt after a *!shell command*.
- `-p` Allows the user to specify a prompt string.
- `-x` Encryption option; when used, `ed` simulates an *X* command and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of *crypt*(1). The *X* command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the `-x` option. See *crypt*(1). Also, see the **WARNINGS** section at the end of this manual page.
- `-C` Encryption option; the same as the `-x` option, except that `ed` simulates a *C* command. The *C* command is like the *X* command, except that all text read in is assumed to have been encrypted.

`ed` operates on a copy of the file it is editing; changes made to the copy have no effect on the file until a *w* (write) command is given. The copy of the text being edited resides in a temporary file called the *buffer*. There is only one buffer.

`red` is a restricted version of `ed`. It will only allow editing of files in the current directory. It prohibits executing shell commands via *!shell command*. Attempts to bypass these restrictions result in an error message (*restricted shell*).

Both `ed` and `red` support the *fspec*(4) formatting capability. After including a format specification as the first line of *file* and invoking `ed` with your terminal in `stty -tabs` or `stty tab3` mode (see *stty*(1)), the specified tab stops will automatically be used when scanning *file*. For example, if the first line of a file contained:

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

tab stops would be set at columns 5, 10, and 15, and a maximum line length of 72 would be imposed. NOTE: when you are entering text into the file, this format is not in effect; instead, because of being in `stty -tabs` or `stty tab3` mode, tabs are expanded to every eighth column.

Commands to `ed` have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command*, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that

the addresses can very often be omitted.

In general, only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While *ed* is accepting text, it is said to be in *input mode*. In this mode, no commands are recognized; all input is merely collected. Leave input mode by typing a period (.) at the beginning of a line, followed immediately by a carriage return.

ed supports a limited form of *regular expression* notation; regular expressions are used in addresses to specify lines and in some commands (e.g., *s*) to specify portions of a line that are to be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE. The REs allowed by *ed* are constructed as follows:

The following *one-character REs* match a *single* character:

- 1.1 An ordinary character (*not* one of those discussed in 1.2 below) is a one-character RE that matches itself.
- 1.2 A backslash (\) followed by any special character is a one-character RE that matches the special character itself. The special characters are:
 - a. ., *, [, and \ (period, asterisk, left square bracket, and backslash, respectively), which are always special, *except* when they appear within square brackets ([] ; see 1.4 below).
 - b. ^ (caret or circumflex), which is special at the *beginning* of an *entire* RE (see 3.1 and 3.2 below), or when it immediately follows the left of a pair of square brackets ([]) (see 1.4 below).
 - c. \$ (dollar sign), which is special at the *end* of an *entire* RE (see 3.2 below).
 - d. The character used to bound (i.e., delimit) an *entire* RE, which is special for that RE (for example, see how slash (/) is used in the *g* command, below.)
- 1.3 A period (.) is a one-character RE that matches any character except new-line.
- 1.4 A non-empty string of characters enclosed in square brackets ([]) is a one-character RE that matches *any one* character in that string. If, however, the first character of the string is a circumflex (^), the one-character RE matches any character *except* new-line and the remaining characters in the string. The ^ has this special meaning *only* if it occurs first in the string. The minus (-) may be used to indicate a range of consecutive ASCII characters; for example, [0-9] is equivalent to [0123456789]. The - loses this special meaning if it occurs first (after an initial ^, if any) or last in the string. The right square bracket (]) does not terminate such a string when it is the first character within it (after an initial ^, if any); e.g., []a-f] matches either a right square bracket (]) or one of the letters a through f inclusive. The four characters listed in 1.2.a above stand for themselves within such a string of characters.

The following rules may be used to construct REs from one-character REs:

- 2.1 A one-character RE is a RE that matches whatever the one-character RE matches.
- 2.2 A one-character RE followed by an asterisk (*) is a RE that matches *zero* or more occurrences of the one-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.
- 2.3 A one-character RE followed by $\{m\}$, $\{m,\}$, or $\{m,n\}$ is a RE that matches a *range* of occurrences of the one-character RE. The values of m and n must be non-negative integers less than 256; $\{m\}$ matches *exactly* m occurrences; $\{m,\}$ matches *at least* m occurrences; $\{m,n\}$ matches *any number* of occurrences *between* m and n inclusive. Whenever a choice exists, the RE matches as many occurrences as possible.
- 2.4 The concatenation of REs is a RE that matches the concatenation of the strings matched by each component of the RE.
- 2.5 A RE enclosed between the character sequences $\{($ and $\}$ is a RE that matches whatever the unadorned RE matches.
- 2.6 The expression $\backslash n$ matches the same string of characters as was matched by an expression enclosed between $\{($ and $\}$ *earlier* in the same RE. Here n is a digit; the sub-expression specified is that beginning with the n -th occurrence of $\{($ counting from the left. For example, the expression $\backslash(.\backslash)\1\$$ matches a line consisting of two repeated appearances of the same string.

Finally, an *entire RE* may be constrained to match only an initial segment or final segment of a line (or both).

- 3.1 A circumflex (^) at the beginning of an entire RE constrains that RE to match an *initial* segment of a line.
- 3.2 A dollar sign (\$) at the end of an entire RE constrains that RE to match a *final* segment of a line.

The construction \wedge *entire RE* $\$$ constrains the entire RE to match the entire line.

The null RE (e.g., $//$) is equivalent to the last RE encountered. See also the last paragraph before FILES below.

To understand addressing in *ed* it is necessary to know that at any time there is a *current line*. Generally speaking, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. *Addresses* are constructed as follows:

1. The character $.$ addresses the current line.
2. The character $\$$ addresses the last line of the buffer.
3. A decimal number n addresses the n -th line of the buffer.
4. $'x$ addresses the line marked with the mark name character x , which must be an ASCII lower-case letter (a-z). Lines are marked with the k command described below.

5. A RE enclosed by slashes (/) addresses the first line found by searching *forward* from the line *following* the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched. See also the last paragraph before FILES below.
6. A RE enclosed in question marks (?) addresses the first line found by searching *backward* from the line *preceding* the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. See also the last paragraph before FILES below.
7. An address followed by a plus sign (+) or a minus sign (-) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. The plus sign may be omitted.
8. If an address begins with + or -, the addition or subtraction is taken with respect to the current line; e.g., -5 is understood to mean .-5.
9. If an address ends with + or -, then 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of Rule 8, immediately above, the address - refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the character ^ in addresses is entirely equivalent to -.) Moreover, trailing + and - characters have a cumulative effect, so -- refers to the current line less 2.
10. For convenience, a comma (,) stands for the address pair 1,\$, while a semicolon (;) stands for the pair .,\$.

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (,). They may also be separated by a semicolon (;). In the latter case, the current line (.) is set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches (see Rules 5 and 6, above). The second address of any two-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of *ed* commands, the default addresses are shown in parentheses. The parentheses are *not* part of the address; they show that the given addresses are the default.

It is generally illegal for more than one command to appear on a line. However, any command (except *e*, *f*, *r*, or *w*) may be suffixed by *l*, *n*, or *p* in which case the current line is either listed, numbered or printed, respectively, as discussed below under the *l*, *n*, and *p* commands.

(.)a
<text>

The *a*ppend command reads the given text and appends it after the addressed line; . is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the "appended" text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.)c
<text>

The *c*hange command deletes the addressed lines, then accepts input text that replaces these lines; . is left at the last line input, or, if there were none, at the first line that was not deleted.

C

Same as the *X* command, except that *ed* assumes all text read in for the *e* and *r* commands is encrypted unless a null key is typed in.

(.,.)d

The *d*elete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

e file

The *e*dit command causes the entire contents of the buffer to be deleted, and then the named file to be read in; . is set to the last line of the buffer. If no file name is given, the currently-remembered file name, if any, is used (see the *f* command). The number of characters read is typed; *file* is remembered for possible use as a default file name in subsequent *e*, *r*, and *w* commands. If *file* is replaced by !, the rest of the line is taken to be a shell (*sh*(1)) command whose output is to be read. Such a shell command is *not* remembered as the current file name. See also **DIAGNOSTICS** below.

E file

The *E*dit command is like *e*, except that the editor does not check to see if any changes have been made to the buffer since the last *w* command.

f file

If *file* is given, the *f*ile-name command changes the currently-remembered file name to *file*; otherwise, it prints the currently-remembered file name.

(1,\$)g/RE/command list

In the global command, the first step is to mark every line that matches the given *RE*. Then, for every such line, the given *command list* is executed with . initially set to that line. A single command or the first of a list of commands appears on the same line as the global

command. All lines of a multi-line list except the last line must be ended with a `\`; `a`, `i`, and `c` commands and associated input are permitted. The `.` terminating input mode may be omitted if it would be the last line of the *command list*. An empty *command list* is equivalent to the `p` command. The `g`, `G`, `v`, and `V` commands are *not* permitted in the *command list*. See also **BUGS** and the last paragraph before **FILES** below.

(1,\$)G/RE/

In the interactive Global command, the first step is to mark every line that matches the given RE. Then, for every such line, that line is printed, `.` is changed to that line, and any *one* command (other than one of the `a`, `c`, `i`, `g`, `G`, `v`, and `V` commands) may be input and is executed. After the execution of that command, the next marked line is printed, and so on; a new-line acts as a null command; an `&` causes the re-execution of the most recent command executed within the current invocation of `G`. Note that the commands input as part of the execution of the `G` command may address and affect *any* lines in the buffer. The `G` command can be terminated by an interrupt signal (ASCII DEL or BREAK).

h

The *help* command gives a short error message that explains the reason for the most recent `?` diagnostic.

H

The *Help* command causes *ed* to enter a mode in which error messages are printed for all subsequent `?` diagnostics. It will also explain the previous `?` if there was one. The `H` command alternately turns this mode on and off; it is initially off.

(.)i

<text>

The *insert* command inserts the given text before the addressed line; `.` is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the `a` command only in the placement of the input text. Address 0 is not legal for this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.,+1)j

The *join* command joins contiguous lines by removing the appropriate new-line characters. If exactly one address is given, this command does nothing.

(.)kx

The *mark* command marks the addressed line with name `x`, which must be an ASCII lower-case letter (`a-z`). The address `'x` then addresses this line; `.` is unchanged.

(..)l

The *list* command prints the addressed lines in an unambiguous way: a few non-printing characters (e.g., *tab*, *backspace*) are represented by

visually mnemonic overstrikes. All other non-printing characters are printed in octal, and long lines are folded. An *l* command may be appended to any other command other than *e*, *f*, *r*, or *w*.

(.,.)*ma*

The *move* command repositions the addressed line(s) after the line addressed by *a*. Address 0 is legal for *a* and causes the addressed line(s) to be moved to the beginning of the file. It is an error if address *a* falls within the range of moved lines; *.* is left at the last line moved.

(.,.)*n*

The *number* command prints the addressed lines, preceding each line by its line number and a tab character; *.* is left at the last line printed. The *n* command may be appended to any other command other than *e*, *f*, *r*, or *w*.

(.,.)*p*

The *print* command prints the addressed lines; *.* is left at the last line printed. The *p* command may be appended to any other command other than *e*, *f*, *r*, or *w*. For example, *dp* deletes the current line and prints the new current line.

P

The editor will prompt with a *** for all subsequent commands. The *P* command alternately turns this mode on and off; it is initially off.

q

The *quit* command causes *ed* to exit. No automatic write of a file is done; however, see **DIAGNOSTICS**, below.

Q

The editor exits without checking if changes have been made in the buffer since the last *w* command.

(**\$**)*r file*

The *read* command reads in the given file after the addressed line. If no file name is given, the currently-remembered file name, if any, is used (see *e* and *f* commands). The currently-remembered file name is *not* changed unless *file* is the very first file name mentioned since *ed* was invoked. Address 0 is legal for *r* and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; *.* is set to the last line read in. If *file* is replaced by *!*, the rest of the line is taken to be a shell (*sh*(1)) command whose output is to be read. For example, "*\$r !ls*" appends current directory to the end of the file being edited. Such a shell command is *not* remembered as the current file name.

(.,.)*s*/RE/*replacement* / or

(.,.)*s*/RE/*replacement* /*g* or

(.,.)*s*/RE/*replacement* /*n* n = 1-512

The *substitute* command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (non-overlapped) matched strings are replaced by the

replacement if the global replacement indicator *g* appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number *n* appears after the command, only the *n*th occurrence of the matched string on each addressed line is replaced. It is an error for the substitution to fail on *all* addressed lines. Any character other than space or new-line may be used instead of / to delimit the RE and the *replacement*; . is left at the last line on which a substitution occurred. See also the last paragraph before FILES below.

An ampersand (&) appearing in the *replacement* is replaced by the string matching the RE on the current line. The special meaning of & in this context may be suppressed by preceding it by \. As a more general feature, the characters \n, where *n* is a digit, are replaced by the text matched by the *n*-th regular subexpression of the specified RE enclosed between \(and \). When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of \(starting from the left. When the character % is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The % loses its special meaning when it is in a replacement string of more than one character or is preceded by a \.

A line may be split by substituting a new-line character into it. The new-line in the *replacement* must be escaped by preceding it by \. Such substitution cannot be done as part of a *g* or *v* command list.

(.,.)*ta*

This command acts just like the *m* command, except that a *copy* of the addressed lines is placed after address *a* (which may be 0); . is left at the last line of the copy.

u

The *undo* command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent *a*, *c*, *d*, *g*, *i*, *j*, *m*, *r*, *s*, *t*, *v*, *G*, or *V* command.

(1,\$)**v**/RE/*command list*

This command is the same as the global command *g* except that the *command list* is executed with . initially set to every line that does *not* match the RE.

(1,\$)**V**/RE/

This command is the same as the interactive global command *G* except that the lines that are marked during the first step are those that do *not* match the RE.

(1,\$)**w** *file*

The *write* command writes the addressed lines into the named file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless your *umask* setting (see *umask*(1)) dictates otherwise. The currently-remembered file name is *not* changed unless *file* is the very first file name mentioned since *ed* was invoked. If no

file name is given, the currently-remembered file name, if any, is used (see *e* and *f* commands); *.* is unchanged. If the command is successful, the number of characters written is typed. If *file* is replaced by *!*, the rest of the line is taken to be a shell (*sh*(1)) command whose standard input is the addressed lines. Such a shell command is *not* remembered as the current file name.

X

A key is prompted for, and it is used in subsequent *e*, *r*, and *w* commands to decrypt and encrypt text using the *crypt*(1) algorithm. An educated guess is made to determine whether text read in for the *e* and *r* commands is encrypted. A null key turns off encryption. Subsequent *e*, *r*, and *w* commands will use this key to encrypt or decrypt the text (see *crypt*(1)). An explicitly empty key turns off encryption. Also, see the *-x* option of *ed*.

(*\$*)-

The line number of the addressed line is typed; *.* is unchanged by this command.

!shell command

The remainder of the line after the *!* is sent to the UNIX system shell (*sh*(1)) to be interpreted as a command. Within the text of that command, the unescaped character *%* is replaced with the remembered file name; if a *!* appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, *!!* will repeat the last shell command. If any expansion is performed, the expanded line is echoed; *.* is unchanged.

(*+*1)<new-line>

An address alone on a line causes the addressed line to be printed. A new-line alone is equivalent to *+.1p*; it is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, *ed* prints a *?* and returns to its command level.

Some size limitations: 512 characters in a line, 256 characters in a global command list, and 64 characters in the pathname of a file (counting slashes). The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

When reading a file, *ed* discards ASCII NUL characters.

If a file is not terminated by a new-line character, *ed* adds one and puts out a message explaining what it did.

If the closing delimiter of a RE or of a replacement string (e.g., */*) would be the last character before a new-line, that delimiter may be omitted, in which case the addressed line is printed. The following pairs of commands are equivalent:

<i>s/s1/s2</i>	<i>s/s1/s2/p</i>
<i>g/s1</i>	<i>g/s1/p</i>
<i>?s1</i>	<i>?s1?</i>

FILES

- \$TMPDIR** if this environmental variable is not null, its value is used in place of **/usr/tmp** as the directory name for the temporary work file.
- /usr/tmp** if **/usr/tmp** exists, it is used as the directory name for the temporary work file.
- /tmp** if the environmental variable **TMPDIR** does not exist or is null, and if **/usr/tmp** does not exist, then **/tmp** is used as the directory name for the temporary work file.
- ed.hup** work is saved here if the terminal is hung up.

NOTES

The **-** option, although it continues to be supported, has been replaced in the documentation by the **-s** option that follows the Command Syntax Standard (see *intro(1)*).

SEE ALSO

edit(1), *ex(1)*, *grep(1)*, *sed(1)*, *sh(1)*, *stty(1)*, *umask(1)*, *vi(1)*, *fspec(4)*, *regexp(5)* in the *System Administrator's Reference Manual*.

DIAGNOSTICS

- ?** for command errors.
- ?file** for an inaccessible file.
(use the *help* and *Help* commands for detailed explanations).

If changes have been made in the buffer since the last *w* command that wrote the entire buffer, *ed* warns the user if an attempt is made to destroy *ed*'s buffer via the *e* or *q* commands. It prints **?** and allows one to continue editing. A second *e* or *q* command at this point will take effect. The **-s** command-line option inhibits this feature.

WARNINGS

The encryption options and commands are provided with the Security Administration Utilities package, which is available only in the United States.

BUGS

- A **!** command cannot be subject to a *g* or a *v* command.
- The **!** command and the **!** escape from the *e*, *r*, and *w* commands cannot be used if the editor is invoked from a restricted shell (see *sh(1)*).
- The sequence **\n** in a RE does not match a new-line character.
- If the editor input is coming from a command file (e.g., *ed file < ed-cmd-file*), the editor will exit at the first failure.

NAME

`edit` – text editor (variant of `ex` for casual users)

SYNOPSIS

`edit [-r] [-x] [-C] name...`

DESCRIPTION

`edit` is a variant of the text editor `ex` recommended for new or casual users who wish to use a command-oriented editor. It operates precisely as `ex(1)` with the following options automatically set:

<code>novice</code>	ON
<code>report</code>	ON
<code>showmode</code>	ON
<code>magic</code>	OFF

These options can be turned on or off via the `set` command in `ex(1)`.

- `-r` Recover file after an editor or system crash.
- `-x` Encryption option; when used the file will be encrypted as it is being written and will require an encryption key to be read. `edit` makes an educated guess to determine if a file is encrypted or not. See `crypt(1)`. Also, see the **WARNING** section at the end of this manual page.
- `-C` Encryption option; the same as `-x` except that `edit` assumes files are encrypted.

The following brief introduction should help you get started with `edit`. If you are using a CRT terminal you may want to learn about the display editor `vi`.

To edit the contents of an existing file you begin with the command `edit name` to the shell. `edit` makes a copy of the file that you can then edit, and tells you how many lines and characters are in the file. To create a new file, you also begin with the command `edit` with a filename: `edit name`; the editor will tell you it is a `[New File]`.

The `edit` command prompt is the colon (:), which you should see after starting the editor. If you are editing an existing file, then you will have some lines in `edit's` buffer (its name for the copy of the file you are editing). When you start editing, `edit` makes the last line of the file the current line. Most commands to `edit` use the current line if you do not tell them which line to use. Thus if you say **print** (which can be abbreviated **p**) and type carriage return (as you should after all `edit` commands), the current line will be printed. If you **delete** (**d**) the current line, `edit` will print the new current line, which is usually the next line in the file. If you **delete** the last line, then the new last line becomes the current one.

If you start with an empty file or wish to add some new lines, then the **append** (**a**) command can be used. After you execute this command (typing a carriage return after the word **append**), `edit` will read lines from your terminal until you type a line consisting of just a dot (.); it places these lines after the current line. The last line you type then becomes the current line. The command **insert** (**i**) is like **append**, but places the lines you type before, rather than after, the current line.

edit numbers the lines in the buffer, with the first line having number 1. If you execute the command **1**, then *edit* will type the first line of the buffer. If you then execute the command **d**, *edit* will delete the first line, line 2 will become line 1, and *edit* will print the current line (the new line 1) so you can see where you are. In general, the current line will always be the last line affected by a command.

You can make a change to some text within the current line by using the **substitute (s)** command: *s/old/new/* where *old* is the string of characters you want to replace and *new* is the string of characters you want to replace *old* with.

The command **file (f)** will tell you how many lines there are in the buffer you are editing and will say [Modified] if you have changed the buffer. After modifying a file, you can save the contents of the file by executing a **write (w)** command. You can leave the editor by issuing a **quit (q)** command. If you run *edit* on a file, but do not change it, it is not necessary (but does no harm) to **write** the file back. If you try to **quit** from *edit* after modifying the buffer without writing it out, you will receive the message `No write since last change (:quit! overrides)`, and *edit* will wait for another command. If you do not want to write the buffer out, issue the **quit** command followed by an exclamation point (**q!**). The buffer is then irretrievably discarded and you return to the shell.

By using the **d** and **a** commands and giving line numbers to see lines in the file, you can make any changes you want. You should learn at least a few more things, however, if you will use *edit* more than a few times.

The **change (c)** command changes the current line to a sequence of lines you supply (as in **append**, you type lines up to a line consisting of only a dot (.). You can tell **change** to change more than one line by giving the line numbers of the lines you want to change, i.e., **3,5c**. You can print lines this way too: **1,23p** prints the first 23 lines of the file.

The **undo (u)** command reverses the effect of the last command you executed that changed the buffer. Thus if you execute a **substitute** command that does not do what you want, type **u** and the old contents of the line will be restored. You can also **undo** an **undo** command. *edit* will give you a warning message when a command affects more than one line of the buffer. Note that commands such as **write** and **quit** cannot be undone.

To look at the next line in the buffer, type carriage return. To look at a number of lines, type **^D** (while holding down the control key, press **d**) rather than carriage return. This will show you a half-screen of lines on a CRT or 12 lines on a hardcopy terminal. You can look at nearby text by executing the **z** command. The current line will appear in the middle of the text displayed, and the last line displayed will become the current line; you can get back to the line where you were before you executed the **z** command by typing **''**. The **z** command has other options: **z-** prints a screen of text (or 24 lines) ending where you are; **z+** prints the next screenful. If you want less than a screenful of lines, type **z.n** to display five lines before and five lines after the current line. (Typing **z.n**, when *n* is an odd number, displays a total of *n* lines, centered about the current line; when *n* is an even number, it displays

$n-1$ lines, so that the lines displayed are centered around the current line.) You can give counts after other commands; for example, you can delete 5 lines starting with the current line with the command **d5**.

To find things in the file, you can use line numbers if you happen to know them; since the line numbers change when you insert and delete lines this is somewhat unreliable. You can search backwards and forwards in the file for strings by giving commands of the form **/text/** to search forward for *text* or **?text?** to search backward for *text*. If a search reaches the end of the file without finding *text*, it wraps around and continues to search back to the line where you are. A useful feature here is a search of the form **/^text/** which searches for *text* at the beginning of a line. Similarly **/text\$/** searches for *text* at the end of a line. You can leave off the trailing **/** or **?** in these commands.

The current line has the symbolic name dot (**.**); this is most useful in a range of lines as in **.,\$p** which prints the current line plus the rest of the lines in the file. To move to the last line in the file, you can refer to it by its symbolic name **\$**. Thus the command **\$d** deletes the last line in the file, no matter what the current line is. Arithmetic with line references is also possible. Thus the line **\$-5** is the fifth before the last and **.+20** is 20 lines after the current line.

You can find out the current line by typing **.=**. This is useful if you wish to move or copy a section of text within a file or between files. Find the first and last line numbers you wish to copy or move. To move lines 10 through 20, type **10,20d a** to delete these lines from the file and place them in a buffer named **a**. *edit* has 26 such buffers named **a** through **z**. To put the contents of buffer **a** after the current line, type **put a**. If you want to move or copy these lines to another file, execute an **edit (e)** command after copying the lines; following the **e** command with the name of the other file you wish to edit, i.e., **edit chapter2**. To copy lines without deleting them, use **yank (y)** in place of **d**. If the text you wish to move or copy is all within one file, it is not necessary to use named buffers. For example, to move lines 10 through 20 to the end of the file, type **10,20m \$**.

SEE ALSO

ed(1), **ex(1)**, **vi(1)**.

WARNING

The encryption options are provided with the Security Administration Utilities package, which is available only in the United States.



NAME

egrep – search a file for a pattern using full regular expressions

SYNOPSIS

egrep [options] full regular expression [file ...]

DESCRIPTION

egrep (*expression grep*) searches files for a pattern of characters and prints all lines that contain that pattern. *egrep* uses full regular expressions (expressions that have string values that use the full set of alphanumeric and special characters) to match the patterns. It uses a fast deterministic algorithm that sometimes needs exponential space.

egrep accepts full regular expressions as in *ed*(1), except for `\(` and `\)`, with the addition of:

1. A full regular expression followed by `+` that matches one or more occurrences of the full regular expression.
2. A full regular expression followed by `?` that matches 0 or 1 occurrences of the full regular expression.
3. Full regular expressions separated by `|` or by a new-line that match strings that are matched by any of the expressions.
4. A full regular expression that may be enclosed in parentheses `()` for grouping.

Be careful using the characters `$`, `*`, `[`, `^`, `|`, `(`, `)`, and `\` in *full regular expression*, because they are also meaningful to the shell. It is safest to enclose the entire *full regular expression* in single quotes `'...'`.

The order of precedence of operators is `[]`, then `*?+`, then concatenation, then `|` and new-line.

If no files are specified, *egrep* assumes standard input. Normally, each line found is copied to the standard output. The file name is printed before each line found if there is more than one input file.

Command line options are:

- b** Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).
- c** Print only a count of the lines that contain the pattern.
- i** Ignore upper/lower case distinction during comparisons.
- l** Print the names of files with matching lines once, separated by new-lines. Does not repeat the names of files when the pattern is found more than once.
- n** Precede each line by its line number in the file (first line is 1).
- v** Print all lines except those that contain the pattern.
- e *special_expression***
Search for a *special expression* (*full regular expression* that begins with a `-`).
- f *file***
Take the list of *full regular expressions* from *file*.

SEE ALSO

ed(1), *fgrep*(1), *grep*(1), *sed*(1), *sh*(1).

DIAGNOSTICS

Exit status is 0 if any matches are found, 1 if none, 2 for syntax errors or inaccessible files (even if matches were found).

BUGS

Ideally there should be only one *grep* command, but there is not a single algorithm that spans a wide enough range of space-time tradeoffs. Lines are limited to BUFSIZ characters; longer lines are truncated. BUFSIZ is defined in `/usr/include/stdio.h`.

NAME

enable, disable – enable/disable LP printers

SYNOPSIS

enable *printers*

disable [*options*] *printers*

DESCRIPTION

The *enable* command activates the named *printers*, enabling them to print requests taken by *lp(1)*. Use *lpstat(1)* to find the status of printers.

The *disable* command deactivates the named *printers*, disabling them from printing requests taken by *lp(1)*. By default, any requests that are currently printing on the designated printers will be reprinted in their entirety either on the same printer or on another member of the same class. Use *lpstat(1)* to find the status of printers. Options for use with *disable* are:

- c** Cancel any requests that are currently printing on any of the designated printers. This option cannot be used with the **-W** option.
- r *reason*** Assign a *reason* for the disabling of the printers. This *reason* applies to all printers mentioned up to the next **-r** option. This *reason* is reported by *lpstat(1)*. If the **-r** option is not present, then a default reason will be used.
- W** Disable the specified printers when the print requests currently printing have finished. This option cannot be used with the **-c** option.

FILES

/usr/spool/lp/*

SEE ALSO

lp(1), *lpstat(1)*.



NAME

`env` – set environment for command execution

SYNOPSIS

`env [-] [name=value] ... [command args]`

DESCRIPTION

`env` obtains the current *environment*, modifies it according to its arguments, then executes the command with the modified environment. Arguments of the form *name=value* are merged into the inherited environment before the command is executed. The `-` flag causes the inherited environment to be ignored completely, so that the command is executed with exactly the environment specified by the arguments.

If no command is specified, the resulting environment is printed, one name-value pair per line.

SEE ALSO

`sh(1)`.

`exec(2)` in the *Programmer's Reference Manual*.

`profile(4)`, `environ(5)` in the *System Administrator's Reference Manual*.



NAME

ex - text editor

SYNOPSIS

ex [-s] [-v] [-t tag] [-r file] [-L] [-R] [-x] [-C] [-c command] file ...

DESCRIPTION

ex is the root of a family of editors: *ex* and *vi*. *ex* is a superset of *ed*, with the most notable extension being a display editing facility. Display based editing is the focus of *vi*.

If you have a CRT terminal, you may wish to use a display based editor; in this case see *vi*(1), which is a command which focuses on the display-editing portion of *ex*.

For *ed* Users

If you have used *ed*(1) you will find that, in addition to having all of the *ed*(1) commands available, *ex* has a number of additional features useful on CRT terminals. Intelligent terminals and high speed terminals are very pleasant to use with *vi*. Generally, the *ex* editor uses far more of the capabilities of terminals than *ed*(1) does, and uses the terminal capability data base (see *terminfo*(4)) and the type of the terminal you are using from the environmental variable *TERM* to determine how to drive your terminal efficiently. The editor makes use of features such as insert and delete character and line in its **visual** command (which can be abbreviated *vi*) and which is the central mode of editing when using *vi*(1).

ex contains a number of features for easily viewing the text of the file. The **z** command gives easy access to windows of text. Typing **^D** (control-d) causes the editor to scroll a half-window of text and is more useful for quickly stepping through a file than just typing return. Of course, the screen-oriented **visual** mode gives constant access to editing context.

ex gives you help when you make mistakes. The **undo** (**u**) command allows you to reverse any single change which goes astray. *ex* gives you a lot of feedback, normally printing changed lines, and indicates when more than a few lines are affected by a command so that it is easy to detect when a command has affected more lines than it should have.

The editor also normally prevents overwriting existing files, unless you edited them, so that you do not accidentally overwrite a file other than the one you are editing. If the system (or editor) crashes, or you accidentally hang up the telephone, you can use the editor **recover** command (or **-r file** option) to retrieve your work. This will get you back to within a few lines of where you left off.

ex has several features for dealing with more than one file at a time. You can give it a list of files on the command line and use the **next** (**n**) command to deal with each in turn. The **next** command can also be given a list of file names, or a pattern as used by the shell to specify a new set of files to be dealt with. In general, file names in the editor may be formed with full shell metasyntax. The metacharacter **'%'** is also available in forming file names and is replaced by the name of the current file.

The editor has a group of buffers whose names are the ASCII lower-case letters (a-z). You can place text in these named buffers where it is available to be inserted elsewhere in the file. The contents of these buffers remain available when you begin editing a new file using the **edit** (e) command.

There is a command **&** in *ex* which repeats the last **substitute** command. In addition, there is a confirmed substitute command. You give a range of substitutions to be done and the editor interactively asks whether each substitution is desired.

It is possible to ignore the case of letters in searches and substitutions. *ex* also allows regular expressions which match words to be constructed. This is convenient, for example, in searching for the word "edit" if your document also contains the word "editor."

ex has a set of options which you can set to tailor it to your liking. One option which is very useful is the **autoindent** option that allows the editor to supply leading white space to align text automatically. You can then use **^D** as a backtab and space or tab to move forward to align new code easily.

Miscellaneous useful features include an intelligent **join** (j) command that supplies white space between joined lines automatically, commands "<" and ">" which shift groups of lines, and the ability to filter portions of the buffer through commands such as *sort*(1).

Invocation Options

The following invocation options are interpreted by *ex* (previously documented options are discussed in the NOTES section at the end of this manual page):

- s Suppress all interactive-user feedback. This is useful in processing editor scripts.
- v Invoke *vi*
- t *tag* Edit the file containing the *tag* and position the editor at its definition.
- r *file* Edit *file* after an editor or system crash. (Recovers the version of *file* that was in the buffer when the crash occurred.)
- L List the names of all files saved as the result of an editor or system crash.
- R **Readonly** mode; the **readonly** flag is set, preventing accidental overwriting of the file.
- x Encryption option; when used, *ex* simulates an X command and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of *crypt*(1). The X command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the -x option. See *crypt*(1). Also, see the WARNINGS section at the end of this manual page.

- C** Encryption option; the same as the **-x** option, except that *ex* simulates a **C** command. The **C** command is like the **X** command, except that all text read in is assumed to have been encrypted.
- c command** Begin editing by executing the specified editor *command* (usually a search or positioning command).

The *file* argument indicates one or more files to be edited.

ex States

- Command** Normal and initial state. Input prompted for by **:**. Your line kill character cancels a partial command.
- Insert** Entered by **a**, **i**, or **c**. Arbitrary text may be entered. Insert state normally is terminated by a line having only "." on it, or, abnormally, with an interrupt.
- Visual** Entered by typing **vi**; terminated by typing **Q** or **^** (control-^).

ex Command Names and Abbreviations

abbrev	ab	map		set	se
append	a	mark	ma	shell	sh
args	ar	move	m	source	so
change	c	next	n	substitute	s
copy	co	number	nu	unabbrev	unab
delete	d	preserve	pre	undo	u
edit	e	print	p	unmap	unm
file	f	put	pu	version	ve
global	g	quit	q	visual	vi
insert	i	read	r	write	w
join	j	recover	rec	xit	x
list	l	rewind	rew	yank	ya

ex Commands

forced encryption	C	heuristic encryption	X
resubst	&	print next	CR
rshift	>	lshift	<
scroll	^D	window	z
shell escape	!		

ex Command Addresses

<i>n</i>	line <i>n</i>	<i>/pat</i>	next with <i>pat</i>
.	current	<i>?pat</i>	previous with <i>pat</i>
\$	last	<i>x-n</i>	<i>n</i> before <i>x</i>
+	next	<i>x,y</i>	<i>x</i> through <i>y</i>
-	previous	<i>'x</i>	marked with <i>x</i>
+n	<i>n</i> forward	"	previous context
%	1, \$		

Initializing options

EXINIT place *set's* here in environment variable
\$HOME/.exrc editor initialization file

<code>./exrc</code>	editor initialization file
<code>set x</code>	enable option <i>x</i>
<code>set nox</code>	disable option <i>x</i>
<code>set x=val</code>	give value <i>val</i> to option <i>x</i>
<code>set</code>	show changed options
<code>set all</code>	show all options
<code>set x?</code>	show value of option <i>x</i>

Most useful options and their abbreviations

<code>autoindent</code>	<code>ai</code>	supply indent
<code>autowrite</code>	<code>aw</code>	write before changing files
<code>directory</code>		pathname of directory for temporary work files
<code>exrc</code>	<code>ex</code>	allow <i>vi/ex</i> to read the <code>.exrc</code> in the current directory. This option is set in the <code>EXINIT</code> shell variable or in the <code>.exrc</code> file in the <code>\$HOME</code> directory.
<code>ignorecase</code>	<code>ic</code>	ignore case of letters in scanning
<code>list</code>		print <code>^I</code> for tab, <code>\$</code> at end
<code>magic</code>		treat <code>.</code> <code>[</code> <code>*</code> special in patterns
<code>modelines</code>		first five lines and last five lines executed as <i>vi/ex</i> commands if they are of the form <code>ex:command:</code> or <code>vi:command:</code>
<code>number</code>	<code>nu</code>	number lines
<code>paragraphs</code>	<code>para</code>	macro names that start paragraphs
<code>redraw</code>		simulate smart terminal
<code>report</code>		informs you if the number of lines modified by the last command is greater than the value of the <code>report</code> variable
<code>scroll</code>		command mode lines
<code>sections</code>	<code>sect</code>	macro names that start sections
<code>shiftwidth</code>	<code>sw</code>	for <code><</code> <code>></code> , and input <code>^D</code>
<code>showmatch</code>	<code>sm</code>	to <code>)</code> and <code>}</code> as typed
<code>showmode</code>	<code>smd</code>	show insert mode in <i>vi</i>
<code>slowopen</code>	<code>slow</code>	stop updates during insert
<code>term</code>		specifies to <i>vi</i> the type of terminal being used (the default is the value of the environmental variable <code>TERM</code>)
<code>window</code>		visual mode lines
<code>wrapmargin</code>	<code>wm</code>	automatic line splitting
<code>wrapscreen</code>	<code>ws</code>	search around end (or beginning) of buffer

Scanning pattern formation

<code>^</code>	beginning of line
<code>\$</code>	end of line
<code>.</code>	any character
<code>\<</code>	beginning of word

\>	end of word
[<i>str</i>]	any character in <i>str</i>
[^ <i>str</i>]	any character not in <i>str</i>
[<i>x-y</i>]	any character between <i>x</i> and <i>y</i>
*	any number of preceding characters

AUTHOR

vi and *ex* are based on software developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

FILES

/usr/lib/exstrings	error messages
/usr/lib/exrecover	recover command
/usr/lib/expreserve	preserve command
/usr/lib/terminfo/*	describes capabilities of terminals
\$HOME/.exrc	editor startup file
./exrc	editor startup file
/tmp/Exnnnnn	editor temporary
/tmp/Rxnnnnn	named buffer temporary
/usr/preserve/login	preservation directory (where <i>login</i> is the user's login)

NOTES

Several options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see *intro(1)*). The *-* option has been replaced by *-s*, a *-r* option that is not followed with an option-argument has been replaced by *-L*, and *+command* has been replaced by *-c command*.

SEE ALSO

crypt(1), *ed(1)*, *edit(1)*, *grep(1)*, *sed(1)*, *sort(1)*, *vi(1)*.
courses(3X), in the *Programmer's Reference Manual*.
term(4), *terminfo(4)* in the *System Administrator's Reference Manual*.
User's Guide.
Editing Guide.
courses/terminfo chapter of the *Programmer's Guide*.

WARNINGS

The encryption options and commands are provided with the Security Administration Utilities package, which is available only in the United States.

BUGS

The *z* command prints the number of logical rather than physical lines. More than a screen full of output may result if long lines are present.

File input/output errors do not print a name if the command line *-s* option is used.

There is no easy way to do a single scan ignoring case.

The editor does not warn if text is placed in named buffers and not used before exiting the editor.

Null characters are discarded in input files and cannot appear in resultant files.



NAME

expr – evaluate arguments as an expression

SYNOPSIS

expr arguments

DESCRIPTION

The arguments are taken as an expression. After evaluation, the result is written on the standard output. Terms of the expression must be separated by blanks. Characters special to the shell must be escaped. Note that 0 is returned to indicate a zero value, rather than the null string. Strings containing blanks or other special characters should be quoted. Integer-valued arguments may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, 2s complement numbers.

The operators and keywords are listed below. Characters that need to be escaped are preceded by \. The list is in order of increasing precedence, with equal precedence operators grouped within {} symbols.

expr \| *expr*

returns the first *expr* if it is neither null nor 0, otherwise returns the second *expr*.

expr \& *expr*

returns the first *expr* if neither *expr* is null or 0, otherwise returns 0.

expr { =, \>, \>=, \<, \<=, != } *expr*

returns the result of an integer comparison if both arguments are integers, otherwise returns the result of a lexical comparison.

expr { +, - } *expr*

addition or subtraction of integer-valued arguments.

expr { *, /, % } *expr*

multiplication, division, or remainder of the integer-valued arguments.

expr : *expr*

The matching operator : compares the first argument with the second argument which must be a regular expression. Regular expression syntax is the same as that of *ed*(1), except that all patterns are “anchored” (i.e., begin with ^) and, therefore, ^ is not a special character, in that context. Normally, the matching operator returns the number of characters matched (0 on failure). Alternatively, the \(...\) pattern symbols can be used to return a portion of the first argument.

EXAMPLES

1. *a*=`*expr* \$*a* + 1`

adds 1 to the shell variable *a*.

2. # `For \$*a* equal to either "/usr/abc/file" or just "file" `
expr \$*a* : `.*\/(.*)` \| \$*a*

returns the last segment of a path name (i.e., file). Watch out for / alone as an argument: *expr* will take it as the division operator (see BUGS below).

3. # A better representation of example 2.

```
expr // $a : '.*\/(.*\)'
```

The addition of the // characters eliminates any ambiguity about the division operator and simplifies the whole expression.

4. expr \$VAR : '.*'

returns the number of characters in \$VAR.

SEE ALSO

ed(1), sh(1).

DIAGNOSTICS

As a side effect of expression evaluation, *expr* returns the following exit values:

- | | |
|---|---|
| 0 | if the expression is neither null nor 0 |
| 1 | if the expression is null or 0 |
| 2 | for invalid expressions. |

syntax error

for operator/operand errors

non-numeric argument

if arithmetic is attempted on such a string

BUGS

After argument processing by the shell, *expr* cannot tell the difference between an operator and an operand except by the value. If \$a is an =, the command:

```
expr $a = '='
```

looks like:

```
expr = = =
```

as the arguments are passed to *expr* (and they will all be taken as the = operator). The following works:

```
expr X$a = X=
```

NAME

factor – obtain the prime factors of a number

SYNOPSIS

factor [integer]

DESCRIPTION

When you use *factor* without an argument, it waits for you to give it an integer. After you give it a positive integer less than or equal to 10^{14} , it factors the integer, prints its prime factors the proper number of times, and then waits for another integer. *factor* exits if it encounters a zero or any non-numeric character.

If you invoke *factor* with an argument, it factors the integer as described above, and then it exits.

The maximum time to factor an integer is proportional to \sqrt{n} . *factor* will take this time when n is prime or the square of a prime.

DIAGNOSTICS

factor prints the error message, "Ouch," for input out of range or for garbage input.



NAME

fgrep – search a file for a character string

SYNOPSIS

fgrep [options] string [file ...]

DESCRIPTION

fgrep (fast *grep*) searches files for a character string and prints all lines that contain that string. *fgrep* is different from *grep(1)* and *egrep(1)* because it searches for a string, instead of searching for a pattern that matches an expression. It uses a fast and compact algorithm.

The characters \$, *, [, ^, |, (,), and \ are interpreted literally by *fgrep*, that is, *fgrep* does not recognize full regular expressions as does *egrep*. Since these characters have special meaning to the shell, it is safest to enclose the entire string in single quotes '...'.

If no files are specified, *fgrep* assumes standard input. Normally, each line found is copied to the standard output. The file name is printed before each line found if there is more than one input file.

Command line options are:

- b Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).
- c Print only a count of the lines that contain the pattern.
- i Ignore upper/lower case distinction during comparisons.
- l Print the names of files with matching lines once, separated by new-lines. Does not repeat the names of files when the pattern is found more than once.
- n Precede each line by its line number in the file (first line is 1).
- v Print all lines except those that contain the pattern.
- x Print only lines matched entirely.
- e *special_string*
Search for a *special string* (*string* begins with a –).
- f *file*
Take the list of *strings* from *file*.

SEE ALSO

ed(1), *egrep(1)*, *grep(1)*, *sed(1)*, *sh(1)*.

DIAGNOSTICS

Exit status is 0 if any matches are found, 1 if none, 2 for syntax errors or inaccessible files (even if matches were found).

BUGS

Ideally there should be only one *grep* command, but there is not a single algorithm that spans a wide enough range of space-time tradeoffs. Lines are limited to BUFSIZ characters; longer lines are truncated. BUFSIZ is defined in */usr/include/stdio.h*.



NAME

file – determine file type

SYNOPSIS

file [**-c**] [**-f** ffile] [**-m** mfile] arg ...

DESCRIPTION

file performs a series of tests on each argument in an attempt to classify it. If an argument appears to be ASCII, *file* examines the first 512 bytes and tries to guess its language. If an argument is an executable **a.out**, *file* will print the version stamp, provided it is greater than 0.

- c** The **-c** option causes *file* to check the magic file for format errors. This validation is not normally carried out for reasons of efficiency. No file typing is done under **-c**.
- f** If the **-f** option is given, the next argument is taken to be a file containing the names of the files to be examined.
- m** The **-m** option instructs *file* to use an alternate magic file.

file uses the file **/etc/magic** to identify files that have some sort of *magic number*, that is, any file containing a numeric or string constant that indicates its type. Commentary at the beginning of **/etc/magic** explains its format.

FILES

/etc/magic

SEE ALSO

filehdr(4) in the *System Administrator's Reference Manual*.



NAME

find – find files

SYNOPSIS

find path-name-list expression

DESCRIPTION

find recursively descends the directory hierarchy for each path name in the *path-name-list* (that is, one or more path names) seeking files that match a boolean *expression* written in the primaries given below. In the descriptions, the argument *n* is used as a decimal integer where *+n* means more than *n*, *-n* means less than *n* and *n* means exactly *n*. Valid expressions are:

- name** *pattern* True if *pattern* matches the current file name. Normal shell file name generation characters (see *sh(1)*) may be used if escaped or quoted.
- [-perm]** *-onum* True if the file permission flags exactly match the octal number *onum* (see *chmod(1)*). If *onum* is prefixed by a minus sign, only the bits that are set in *onum* are compared with the file permission flags, and the expression evaluates true if they match.
- type** *c* True if the type of the file is *c*, where *c* is *b*, *c*, *d*, *p*, or *f* for block special file, character special file, directory, fifo (a.k.a. named pipe), or plain file respectively.
- links** *n* True if the file has *n* links.
- user** *uname* True if the file belongs to the user *uname*. If *uname* is numeric and does not appear as a login name in the */etc/passwd* file, it is taken as a user ID.
- group** *gname* True if the file belongs to the group *gname*. If *gname* is numeric and does not appear in the */etc/group* file, it is taken as a group ID.
- size** *n*[*c*] True if the file is *n* blocks long (512 bytes per block). If *n* is followed by a *c*, the size is in characters.
- atime** *n* True if the file has been accessed in *n* days. The access time of directories in *path-name-list* is changed by *find* itself.
- mtime** *n* True if the file has been modified in *n* days.
- ctime** *n* True if the file has been changed in *n* days.
- exec** *cmd* True if the executed *cmd* returns a zero value as exit status. The end of *cmd* must be punctuated by an escaped semi-colon. A command argument *{* is replaced by the current path name.
- ok** *cmd* Like **-exec** except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing *y*.
- print** Always true; causes the current path name to be printed.

- cpio *device*** Always true; write the current file on *device* in *cpio*(1) format (5120-byte records).
- newer *file*** True if the current file has been modified more recently than the argument *file*.
- depth** Always true; causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when *find* is used with *cpio*(1) to transfer files that are contained in directories without write permission.
- mount** Always true; restricts the search to the file system containing the directory specified, or if no directory was specified, the current directory.
- local** True if the file physically resides on the local system.
- (*expression*)** True if the parenthesized expression is true (parentheses are special to the shell and must be escaped).

The primaries may be combined using the following operators (in order of decreasing precedence):

- 1) The negation of a primary (! is the unary *not* operator).
- 2) Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries).
- 3) Alternation of primaries (-o is the *or* operator).

EXAMPLE

To remove all files named **a.out** or ***.o** that have not been accessed for a week:

```
find / \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

FILES

/etc/passwd, /etc/group

SEE ALSO

chmod(1), *cpio*(1), *sh*(1), *test*(1).
stat(2), *umask*(2) in the *Programmer's Reference Manual*.
fs(4) in the *System Administrator's Reference Manual*.

BUGS

find / -depth always fails with the message: "find: stat failed: : No such file or directory".

NAME

getopt – parse command options

SYNOPSIS

```
set -- `getopt optstring $*`
```

DESCRIPTION

WARNING: Start using the new command *getopts(1)* in place of *getopt(1)*. *getopt(1)* will not be supported in the next major release. For more information, see the **WARNINGS** section, below.

getopt is used to break up options in command lines for easy parsing by shell procedures and to check for legal options. *optstring* is a string of recognized option letters (see *getopt(3C)*); if a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by white space. The special option `--` is used to delimit the end of the options. If it is used explicitly, *getopt* will recognize it; otherwise, *getopt* will generate it; in either case, *getopt* will place it at the end of the options. The positional parameters (`$1 $2 ...`) of the shell are reset so that each option is preceded by a `-` and is in its own positional parameter; each option argument is also parsed into its own positional parameter.

EXAMPLE

The following code fragment shows how one might process the arguments for a command that can take the options `a` or `b`, as well as the option `o`, which requires an argument:

```
set -- `getopt abo: $*`
if [ $? != 0 ]
then
    echo $USAGE
    exit 2
fi
for i in $*
do
    case $i in
    -a | -b) FLAG=$i; shift;;
    -o)      OARG=$2; shift 2;;
    --)     shift; break;;
    esac
done
```

This code will accept any of the following as equivalent:

```
cmd -aoarg file file
cmd -a -o arg file file
cmd -oarg -a file file
cmd -a -oarg -- file file
```

SEE ALSO

getopts(1), *sh(1)*.
getopt(3C) in the *Programmer's Reference Manual*.

DIAGNOSTICS

getopt prints an error message on the standard error when it encounters an option letter not included in *optstring*.

WARNINGS

getopt(1) does not support the part of Rule 8 of the command syntax standard (see *intro*(1)) that permits groups of option-arguments following an option to be separated by white space and quoted. For example,

```
cmd -a -b -o "xxx z yy" file
```

is not handled correctly). To correct this deficiency, use the new command *getopts*(1) in place of *getopt*(1).

getopt(1) will not be supported in the next major release. For this release a conversion tool has been provided, *getoptcvt*. For more information about *getopts* and *getoptcvt*, see the *getopts*(1) manual page.

If an option that takes an option-argument is followed by a value that is the same as one of the options listed in *optstring* (referring to the earlier EXAMPLE section, but using the following command line: `cmd -o -a file`), *getopt* will always treat `-a` as an option-argument to `-o`; it will never recognize `-a` as an option. For this case, the `for` loop in the example will shift past the *file* argument.

NAME

`getopts`, `getoptcv` – parse command options

SYNOPSIS

`getopts` *optstring* *name* [*arg* ...]

`/usr/lib/getoptcv` [`-b`] *file*

DESCRIPTION

`getopts` is used by shell procedures to parse positional parameters and to check for legal options. It supports all applicable rules of the command syntax standard (see Rules 3-10, *intro*(1)). It should be used in place of the `getopt`(1) command. (See the **WARNING**, below.)

optstring must contain the option letters the command using `getopts` will recognize; if a letter is followed by a colon, the option is expected to have an argument, or group of arguments, which must be separated from it by white space.

Each time it is invoked, `getopts` will place the next option in the shell variable *name* and the index of the next argument to be processed in the shell variable **OPTIND**. Whenever the shell or a shell procedure is invoked, **OPTIND** is initialized to 1.

When an option requires an option-argument, `getopts` places it in the shell variable **OPTARG**.

If an illegal option is encountered, `?` will be placed in *name*.

When the end of options is encountered, `getopts` exits with a non-zero exit status. The special option “`---`” may be used to delimit the end of the options.

By default, `getopts` parses the positional parameters. If extra arguments (*arg* ...) are given on the `getopts` command line, `getopts` will parse them instead.

`/usr/lib/getoptcv` reads the shell script in *file*, converts it to use `getopts`(1) instead of `getopt`(1), and writes the results on the standard output.

`-b` the results of running `/usr/lib/getoptcv` will be portable to earlier releases of the UNIX system. `/usr/lib/getoptcv` modifies the shell script in *file* so that when the resulting shell script is executed, it determines at run time whether to invoke `getopts`(1) or `getopt`(1).

So all new commands will adhere to the command syntax standard described in *intro*(1), they should use `getopts`(1) or `getopt`(3C) to parse positional parameters and check for options that are legal for that command (see **WARNINGS**, below).

EXAMPLE

The following fragment of a shell program shows how one might process the arguments for a command that can take the options `a` or `b`, as well as the option `o`, which requires an option-argument:

```
while getopts abo: c
do
    case $c in
        a | b)    FLAG=$c;;
```

```

o)          OARG=$OPTARG;;
\?)        echo $USAGE
          exit 2;;

esac

done
shift `expr $OPTIND - 1`

```

This code will accept any of the following as equivalent:

```

cmd -a -b -o "xxx z yy" file
cmd -a -b -o "xxx z yy" -- file
cmd -ab -o xxx,z,yy file
cmd -ab -o "xxx z yy" file
cmd -o xxx,z,yy -b -a file

```

SEE ALSO

intro(1), sh(1).
getopts(3C) in the *Programmer's Reference Manual*.
UNIX System V Release 3.0 Release Notes.

WARNING

Although the following command syntax rule (see *intro(1)*) relaxations are permitted under the current implementation, they should not be used because they may not be supported in future releases of the system. As in the EXAMPLE section above, **a** and **b** are options, and the option **o** requires an option-argument:

```

cmd -abxxx file (Rule 5 violation: options with
option-arguments must not be grouped with other options)
cmd -ab -oxxx file (Rule 6 violation: there must be
white space after an option that takes an option-argument)

```

Changing the value of the shell variable **OPTIND** or parsing different sets of arguments may lead to unexpected results.

DIAGNOSTICS

getopts prints an error message on the standard error when it encounters an option letter not included in *optstring*.

NAME

graph – draw a graph

SYNOPSIS

graph [options]

DESCRIPTION

graph with no options takes pairs of numbers from the standard input as abscissas and ordinates of a graph. Successive points are connected by straight lines. The graph is encoded on the standard output for display by the *tplot*(1G) filters.

If the coordinates of a point are followed by a non-numeric string, that string is printed as a label beginning on the point. Labels may be surrounded with quotes ", in which case they may be empty or contain blanks and numbers; labels never contain new-lines.

The following options are recognized, each as a separate argument:

- a Supply abscissas automatically (they are missing from the input); spacing is given by the next argument (default 1). A second optional argument is the starting point for automatic abscissas (default 0 or lower limit given by –x).
- b Break (disconnect) the graph after each label in the input.
- c Character string given by next argument is default label for each point.
- g Next argument is grid style, 0 no grid, 1 frame with ticks, 2 full grid (default).
- l Next argument is label for graph.
- m Next argument is mode (style) of connecting lines: 0 disconnected, 1 connected (default). Some devices give distinguishable line styles for other small integers (e.g., the Tektronix 4014: 2=dotted, 3=dash-dot, 4=short-dash, 5=long-dash).
- s Save screen, do not erase before plotting.
- x [1] If 1 is present, x axis is logarithmic. Next 1 (or 2) arguments are lower (and upper) x limits. Third argument, if present, is grid spacing on x axis. Normally these quantities are determined automatically.
- y [1] Similarly for y.
- h Next argument is fraction of space for height.
- w Similarly for width.
- r Next argument is fraction of space to move right before plotting.
- u Similarly to move up before plotting.
- t Transpose horizontal and vertical axes. (Option –x now applies to the vertical axis.)

A legend indicating grid range is produced with a grid unless the –s option is present. If a specified lower limit exceeds the upper limit, the axis is reversed.

SEE ALSO

graphics(1G), spline(1G), tplot(1G).

BUGS

graph stores all points internally and drops those for which there is no room. Segments that run out of bounds are dropped, not windowed. Logarithmic axes may not be reversed.

NAME

grep – search a file for a pattern

SYNOPSIS

grep [options] limited regular expression [file ...]

DESCRIPTION

grep searches files for a pattern and prints all lines that contain that pattern. *grep* uses limited regular expressions (expressions that have string values that use a subset of the possible alphanumeric and special characters) like those used with *ed* (1) to match the patterns. It uses a compact non-deterministic algorithm.

Be careful using the characters \$, *, [, ^, |, (,), and \ in the *limited regular expression* because they are also meaningful to the shell. It is safest to enclose the entire *limited regular expression* in single quotes '...'

If no files are specified, *grep* assumes standard input. Normally, each line found is copied to standard output. The file name is printed before each line found if there is more than one input file.

Command line options are:

- b Precede each line by the block number on which it was found. This can be useful in locating block numbers by context (first block is 0).
- c Print only a count of the lines that contain the pattern.
- i Ignore upper/lower case distinction during comparisons.
- l Print the names of files with matching lines once, separated by newlines. Does not repeat the names of files when the pattern is found more than once.
- n Precede each line by its line number in the file (first line is 1).
- s Suppress error messages about nonexistent or unreadable files
- v Print all lines except those that contain the pattern.

SEE ALSO

ed(1), egrep(1), fgrep(1), sed(1), sh(1).

DIAGNOSTICS

Exit status is 0 if any matches are found, 1 if none, 2 for syntax errors or inaccessible files (even if matches were found).

BUGS

Lines are limited to BUFSIZ characters; longer lines are truncated. BUFSIZ is defined in `/usr/include/stdio.h`.

If there is a line with embedded nulls, *grep* will only match up to the first null; if it matches, it will print the entire line.



NAME

id – print user and group IDs and names

SYNOPSIS

id

DESCRIPTION

id outputs the user and group IDs and the corresponding names of the invoking process. If the effective and real IDs are different, both are printed.

SEE ALSO

logname(1) in the *User's Reference Manual*.
getuid(2) in the *Programmer's Reference Manual*.



NAME

ipcrm – remove a message queue, semaphore set or shared memory id

SYNOPSIS

ipcrm [*options*]

DESCRIPTION

ipcrm will remove one or more specified messages, semaphore or shared memory identifiers. The identifiers are specified by the following *options*:

- q** *msqid* removes the message queue identifier *msqid* from the system and destroys the message queue and data structure associated with it.
- m** *shmid* removes the shared memory identifier *shmid* from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.
- s** *semid* removes the semaphore identifier *semid* from the system and destroys the set of semaphores and data structure associated with it.
- Q** *msgkey* removes the message queue identifier, created with key *msgkey*, from the system and destroys the message queue and data structure associated with it.
- M** *shmkey* removes the shared memory identifier, created with key *shmkey*, from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.
- S** *semkey* removes the semaphore identifier, created with key *semkey*, from the system and destroys the set of semaphores and data structure associated with it.

The details of the removes are described in *msgctl(2)*, *shmctl(2)*, and *semctl(2)*. The identifiers and keys may be found by using *ipcs(1)*.

SEE ALSO

ipcs(1),
msgctl(2), *msgget(2)*, *msgop(2)*, *semctl(2)*, *semget(2)*, *semop(2)*, *shmctl(2)*,
shmget(2), *shmop(2)* in the *Programmer's Reference Manual*.



NAME

`ipcs` – report inter-process communication facilities status

SYNOPSIS

`ipcs` [options]

DESCRIPTION

`ipcs` prints certain information about active inter-process communication facilities. Without *options*, information is printed in short format for message queues, shared memory, and semaphores that are currently active in the system. Otherwise, the information that is displayed is controlled by the following *options*:

- `-q` Print information about active message queues.
- `-m` Print information about active shared memory segments.
- `-s` Print information about active semaphores.

If any of the options `-q`, `-m`, or `-s` are specified, information about only those indicated will be printed. If none of these three are specified, information about all three will be printed subject to these options:

- `-b` Print biggest allowable size information. (Maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set for semaphores.) See below for meaning of columns in a listing.
- `-c` Print creator's login name and group name. See below.
- `-o` Print information on outstanding usage. (Number of messages on queue and total number of bytes in messages on queue for message queues and number of processes attached to shared memory segments.)
- `-p` Print process number information. (Process ID of last process to send a message and process ID of last process to receive a message on message queues and process ID of creating process and process ID of last process to attach or detach on shared memory segments) See below.
- `-t` Print time information. (Time of the last control operation that changed the access permissions for all facilities. Time of last `msgsnd` and last `msgrcv` on message queues, last `shmat` and last `shmdt` on shared memory, last `semop(2)` on semaphores.) See below.
- `-a` Use all print *options*. (This is a shorthand notation for `-b`, `-c`, `-o`, `-p`, and `-t`.)
- `-C corefile`
Use the file *corefile* in place of `/dev/kmem`.
- `-N namelist`
Use the file *namelist* in place of `/unix`.

The column headings and the meaning of the columns in an `ipcs` listing are given below; the letters in parentheses indicate the *options* that cause the

corresponding heading to appear; **all** means that the heading always appears. Note that these *options* only determine what information is provided for each facility; they do *not* determine which facilities will be listed.

- T** (all) Type of the facility:
- q** message queue;
 - m** shared memory segment;
 - s** semaphore.
- ID** (all) The identifier for the facility entry.
- KEY** (all) The key used as an argument to *msgget*, *semget*, or *shmget* to create the facility entry. (Note: The key of a shared memory segment is changed to `IPC_PRIVATE` when the segment has been removed until all processes attached to the segment detach it.)
- MODE** (all) The facility access modes and flags: The mode consists of 11 characters that are interpreted as follows:
The first two characters are:
- R** if a process is waiting on a *msgrcv*;
 - S** if a process is waiting on a *msgsnd*;
 - D** if the associated shared memory segment has been removed. It will disappear when the last process attached to the segment detaches it;
 - C** if the associated shared memory segment is to be cleared when the first attach is executed;
 - if the corresponding special flag is not set.
- The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.
- The permissions are indicated as follows:
- r** if read permission is granted;
 - w** if write permission is granted;
 - a** if alter permission is granted;
 - if the indicated permission is *not* granted.
- OWNER** (all) The login name of the owner of the facility entry.
- GROUP** (all) The group name of the group of the owner of the facility entry.
- CREATOR** (a,c) The login name of the creator of the facility entry.

CGROUP	(a,c)	The group name of the group of the creator of the facility entry.
CBYTES	(a,o)	The number of bytes in messages currently outstanding on the associated message queue.
QNUM	(a,o)	The number of messages currently outstanding on the associated message queue.
QBYTES	(a,b)	The maximum number of bytes allowed in messages outstanding on the associated message queue.
LSPID	(a,p)	The process ID of the last process to send a message to the associated queue.
LRPID	(a,p)	The process ID of the last process to receive a message from the associated queue.
STIME	(a,t)	The time the last message was sent to the associated queue.
RTIME	(a,t)	The time the last message was received from the associated queue.
CTIME	(a,t)	The time when the associated entry was created or changed.
NATTCH	(a,o)	The number of processes attached to the associated shared memory segment.
SEGSZ	(a,b)	The size of the associated shared memory segment.
CPID	(a,p)	The process ID of the creator of the shared memory entry.
LPID	(a,p)	The process ID of the last process to attach or detach the shared memory segment.
ATIME	(a,t)	The time the last attach was completed to the associated shared memory segment.
DTIME	(a,t)	The time the last detach was completed on the associated shared memory segment.
NSEMS	(a,b)	The number of semaphores in the set associated with the semaphore entry.
OTIME	(a,t)	The time the last semaphore operation was completed on the set associated with the semaphore entry.

FILES

/unix	system namelist
/dev/kmem	memory
/etc/passwd	user names
/etc/group	group names

WARNING

If the user specifies either the `-C` or `-N` flag, the real and effective UID/GID will be set to the real UID/GID of the user invoking *ipcs*.

SEE ALSO

msgop(2), *semop(2)*, *shmop(2)* in the *Programmer's Reference Manual*.

BUGS

Things can change while *ipcs* is running; the picture it gives is only a close approximation to reality.

NAME

`ismpx` - return windowing terminal state

SYNOPSIS

`ismpx [-s]`

DESCRIPTION

The *ismpx* command reports whether its standard input is connected to a multiplexed *xt(7)* channel; i.e., whether it's running under *layers(1)* or not. It is useful for shell scripts that download programs to a windowing terminal or depend on screen size.

ismpx prints **yes** and returns 0 if invoked under *layers(1)*, and prints **no** and returns 1 otherwise.

`-s` Do not print anything; just return the proper exit status.

EXIT STATUS

Returns 0 if invoked under *layers(1)*, 1 if not.

SEE ALSO

layers(1), *jwin(1)*.

xt(7) in the *System Administrator's Reference Manual*.

EXAMPLE

```
if ismpx -s
then
    jwin
fi
```



NAME

join – relational database operator

SYNOPSIS

join [options] file1 file2

DESCRIPTION

join forms, on the standard output, a join of the two relations specified by the lines of *file1* and *file2*. If *file1* is *-*, the standard input is used.

File1 and *file2* must be sorted in increasing ASCII collating sequence on the fields on which they are to be joined, normally the first in each line [see *sort(1)*].

There is one line in the output for each pair of lines in *file1* and *file2* that have identical join fields. The output line normally consists of the common field, then the rest of the line from *file1*, then the rest of the line from *file2*.

The default input field separators are blank, tab, or new-line. In this case, multiple separators count as one field separator, and leading separators are ignored. The default output field separator is a blank.

Some of the below options use the argument *n*. This argument should be a 1 or a 2 referring to either *file1* or *file2*, respectively. The following options are recognized:

- an** In addition to the normal output, produce a line for each unpairable line in file *n*, where *n* is 1 or 2.
- e s** Replace empty output fields by string *s*.
- jn m** Join on the *m*th field of file *n*. If *n* is missing, use the *m*th field in each file. Fields are numbered starting with 1.
- o list** Each output line comprises the fields specified in *list*, each element of which has the form *n.m*, where *n* is a file number and *m* is a field number. The common field is not printed unless specifically requested.
- tc** Use character *c* as a separator (tab character). Every appearance of *c* in a line is significant. The character *c* is used as the field separator for both input and output.

EXAMPLE

The following command line will join the password file and the group file, matching on the numeric group ID, and outputting the login name, the group name and the login directory. It is assumed that the files have been sorted in ASCII collating sequence on the group ID fields.

```
join -j1 4 -j2 3 -o 1.1 2.1 1.6 -t: /etc/passwd /etc/group
```

SEE ALSO

awk(1), comm(1), sort(1), uniq(1).

BUGS

With default field separation, the collating sequence is that of **sort -b**; with **-t**, the sequence is that of a plain sort.

The conventions of *join*, *sort*, *comm*, *uniq* and *awk(1)* are wildly incongruous. Filenames that are numeric may cause conflict when the **-o** option is used right before listing filenames.

NAME

jterm – reset layer of windowing terminal

SYNOPSIS

jterm

DESCRIPTION

The *jterm* command is used to reset a layer of a windowing terminal after downloading a terminal program that changes the terminal attributes of the layer. It is useful only under *layers(1)*. In practice, it is most commonly used to restart the default terminal emulator after using an alternate one provided with a terminal-specific application package. For example, on the AT&T Teletype 5620 DMD terminal, after executing the *hp2621(1)* command in a layer, issuing the *jterm* command will restart the default terminal emulator in that layer.

EXIT STATUS

Returns 0 upon successful completion, 1 otherwise.

NOTE

The layer that is reset is the one attached to standard error; that is, the window you are in when you type the *jterm* command.

SEE ALSO

layers(1).



NAME

`jwin` – print size of layer

SYNOPSIS

`jwin`

DESCRIPTION

`jwin` runs only under *layers*(1) and is used to determine the size of the layer associated with the current process. It prints the width and the height of the layer in bytes (number of characters across and number of lines, respectively). For bit-mapped terminals only, it also prints the width and height of the layer in bits.

EXIT STATUS

Returns 0 on successful completion, 1 otherwise.

DIAGNOSTICS

If *layers*(1) has not been invoked, an error message is printed:

```
jwin: not mpx
```

NOTE

The layer whose size is printed is the one attached to standard input; that is, the window you are in when you type the `jwin` command.

SEE ALSO

layers(1).

EXAMPLE

```
jwin
bytes:  86 25
bits:   780 406
```



NAME

kill – terminate a process

SYNOPSIS

kill [-signo] PID ...

DESCRIPTION

kill sends signal 15 (terminate) to the specified processes. This will normally kill processes that do not catch or ignore the signal. The process number of each asynchronous process started with **&** is reported by the shell (unless more than one process is started in a pipeline, in which case the number of the last process in the pipeline is reported). Process numbers can also be found by using *ps*(1).

The details of the kill are described in *kill*(2). For example, if process number 0 is specified, all processes in the process group are signaled.

The killed process must belong to the current user unless he is the super-user.

If a signal number preceded by **-** is given as first argument, that signal is sent instead of terminate (see *signal*(2)). In particular “kill -9 ...” is a sure kill.

SEE ALSO

ps(1), *sh*(1).

kill(2), *signal*(2) in the *Programmer's Reference Manual*.



NAME

layers – layer multiplexor for windowing terminals

SYNOPSIS

layers [-s] [-t] [-d] [-p] [-f file] [*layersys-prgm*]

DESCRIPTION

layers manages asynchronous windows (see *layers(5)*) on a windowing terminal. Upon invocation, *layers* finds an unused *xt(7)* channel group and associates it with the terminal line on its standard output. It then waits for commands from the terminal.

Command-line options:

- s Reports protocol statistics on standard error at the end of the session after you exit from *layers*. The statistics may be printed during a session by invoking the program *xts(1M)*.
- t Turns on *xt(7)* driver packet tracing, and produces a trace dump on standard error at the end of the session after you exit from *layers*. The trace dump may be printed during a session by invoking the program *xtt(1M)*.
- d If a firmware patch has been downloaded, prints out the sizes of the text, data, and bss portions of the firmware patch on standard error.
- p If a firmware patch has been downloaded, prints the downloading protocol statistics and a trace on standard error.
- f *file* Starts *layers* with an initial configuration specified by *file*. Each line of the file represents a layer to be created, and has the following format:

```
origin_x origin_y corner_x corner_y command_list
```

The coordinates specify the size and position of the layer on the screen in the terminal's coordinate system. If all four are 0, the user must define the layer interactively. *command_list*, a list of one or more commands, must be provided. It is executed in the newlayer using the user's shell (by executing: `$SHELL -i -c "command_list"`). This means that the last command should invoke a shell, such as `/bin/sh`. (If the last command is not a shell, then, when the last command has completed, the layer will not be functional.)

layersys-prgm

A file containing a firmware patch that the *layers* command downloads to the terminal before layers are created and *command_list* is executed.

Each layer is in most ways functionally identical to a separate terminal. Characters typed on the keyboard are sent to the standard input of the UNIX system process attached to the current layer (called the host process), and characters written on the standard output by the host process appear in that layer. When a layer is created, a separate shell is established and bound to the layer. If the environment variable SHELL is set, the user will get that shell:

otherwise, */bin/sh* will be used. In order to enable communications with other users via *write(1)*, *layers* invokes the command *relogin(1M)* when the first layer is created. *relogin(1M)* will reassign that layer as the user's logged-in terminal. An alternative layer can be designated by using *relogin(1M)* directly. *layers* will restore the original assignment on termination.

Layers are created, deleted, reshaped, and otherwise manipulated in a terminal-dependent manner. For instance, the AT&T Teletype 5620 DMD terminal provides a mouse-activated pop-up menu of layer operations. The method of ending a *layers* session is also defined by the terminal.

If a user wishes to take advantage of a terminal-specific application software package, the environment variable *DMD* should be set to the pathname of the directory where the package was installed. Otherwise *DMD* should not be set.

EXAMPLE

```
layers -f startup
```

where *startup* contains

```
8 8 700 200 date ; pwd ; exec $SHELL
8 300 780 850 exec $SHELL
```

NOTES

The *xt(7)* driver supports an alternate data transmission scheme known as ENCODING MODE. This mode makes *layers* operation possible even over data links which intercept control characters or do not transmit 8-bit characters. ENCODING MODE is selected either by setting a configuration option on your windowing terminal or by setting the environment variable *DMDLOAD* to the value *hex* before running *layers*:

```
export DMDLOAD; DMDLOAD=hex
```

If, after executing *layers -f file*, the terminal does not respond in one or more of the layers, often the last command in the *command-list* for that layer did not invoke a shell.

WARNING

To access this version of *layers*, make sure */usr/bin* appears before any other directory, such as *\$DMD/bin*, you have in your path that contains a *layers* program. (For information about defining the shell environmental variable *PATH* in your *.profile*, see *profile(4)*) Otherwise, if there is a terminal-dependent version of *layers*, you may get it instead of the correct one.

When invoking *layers* with the *-s*, *-t*, *-d*, or *-p* options, it is best to redirect standard error to another file to save the statistics and tracing output (e.g., *layers -s 2>stats*); otherwise all or some of the output may be lost.

FILES

```
/dev/xt??[0-7]
/usr/lib/layersys/lsys.8;7;3
$DMD/lib/layersys/lsys.8;?;?
```

SEE ALSO

sh(1), write(1).

libwindows(3X) in the *Programmer's Reference Manual*.

relogin(1M), xt(7), xts(1M), xtt(1M), wtinit(1M), layers(5) in the *System Administrator's Reference Manual*.



LINE(1)

(User Environment Utilities)

LINE(1)

NAME

line – read one line

SYNOPSIS

line

DESCRIPTION

line copies one line (up to a new-line) from the standard input and writes it on the standard output. It returns an exit code of 1 on EOF and always prints at least a new-line. It is often used within shell files to read from the user's terminal.

SEE ALSO

sh(1).
read(2) in the *Programmer's Reference Manual*.



NAME

login – sign on

SYNOPSIS

login [name [env-var ...]]

DESCRIPTION

The *login* command is used at the beginning of each terminal session and allows you to identify yourself to the system. It may be invoked as a command or by the system when a connection is first established. Also, it is invoked by the system when a previous user has terminated the initial shell by typing a *cntrl-d* to indicate an "end-of-file."

If *login* is invoked as a command it must replace the initial command interpreter. This is accomplished by typing:

```
exec login
from the initial shell.
```

login asks for your user name (if not supplied as an argument), and, if appropriate, your password. Echoing is turned off (where possible) during the typing of your password, so it will not appear on the written record of the session.

If you make any mistake in the login procedure you will receive the message

Login incorrect

and a new login prompt will appear. If you make five incorrect login attempts, all five may be logged in */usr/adm/loginlog* (if it exists) and the line will be dropped.

If you do not complete the login successfully within a certain period of time (e.g., one minute), you are likely to be silently disconnected.

After a successful login, accounting files are updated, the procedure */etc/profile* is performed, the message-of-the-day, if any, is printed, the user-ID, the group-ID, the working directory, and the command interpreter (usually *sh*(1)) is initialized, and the file *.profile* in the working directory is executed, if it exists. These specifications are found in the */etc/passwd* file entry for the user. The name of the command interpreter is – followed by the last component of the interpreter's path name (i.e., *-sh*). If this field in the password file is empty, then the default command interpreter, */bin/sh* is used. If this field is *, then the named directory becomes the root directory, the starting point for path searches for path names beginning with a /. At that point *login* is re-executed at the new level which must have its own root structure, including */etc/login* and */etc/passwd*.

The basic *environment* is initialized to:

```
HOME=your-login-directory
PATH=:/bin:/usr/bin
SHELL=last-field-of-passwd-entry
MAIL=/usr/mail/your-login-name
TZ=timezone-specification
```

The environment may be expanded or modified by supplying additional arguments to *login*, either at execution time or when *login* requests your login

name. The arguments may take either the form *xxx* or *xxx=yyy*. Arguments without an equal sign are placed in the environment as

`Ln=xxx`

where *n* is a number starting at 0 and is incremented each time a new variable name is required. Variables containing an = are placed into the environment without modification. If they already appear in the environment, then they replace the older value. There are two exceptions. The variables PATH and SHELL cannot be changed. This prevents people, logging into restricted shell environments, from spawning secondary shells which are not restricted. Both *login* and *getty* understand simple single-character quoting conventions. Typing a backslash in front of a character quotes it and allows the inclusion of such things as spaces and tabs.

FILES

<code>/etc/utmp</code>	accounting
<code>/etc/wtmp</code>	accounting
<code>/usr/mail/<i>your-name</i></code>	mailbox for user <i>your-name</i>
<code>/usr/adm/loginlog</code>	record of failed login attempts
<code>/etc/motd</code>	message-of-the-day
<code>/etc/passwd</code>	password file
<code>/etc/profile</code>	system profile
<code>.profile</code>	user's login profile

SEE ALSO

`mail(1)`, `newgrp(1)`, `sh(1)`, `su(1M)`.
`loginlog(4)`, `passwd(4)`, `profile(4)`, `environ(5)` in the *System Administrator's Reference Manual*.

DIAGNOSTICS

login incorrect if the user name or the password cannot be matched.
No shell, cannot open password file, or no directory: consult a UNIX system programming counselor.
No utmp entry. You must exec "login" from the lowest level "sh" if you attempted to execute login as a command without using the shell's exec internal command or from other than the initial shell.

NAME

logname – get login name

SYNOPSIS

logname

DESCRIPTION

logname returns the contents of the environment variable **\$LOGNAME**, which is set when a user logs into the system.

FILES

/etc/profile

SEE ALSO

env(1), login(1).

logname(3X) in the *Programmer's Reference Manual*.

environ(5) in the *System Administrator's Reference Manual*.



NAME

lp, *cancel* – send/cancel requests to an LP print service

SYNOPSIS

lp [*printing-options*] *files*

lp **-i** *request-ids* *printing-options*

cancel [*request-ids*] [*printers*]

DESCRIPTION

The first form of the *lp* shell command arranges for the named files and associated information (collectively called a *request*) to be printed. If no file names are specified on the shell command line, the standard input is assumed. The standard input may be specified along with named *files* on the shell command line by using the file name(s) and **-** for the standard input. The *files* will be printed in the order they appear on the shell command line.

The second form of *lp* is used to change the options for a request. The print request identified by the *request-id* is changed according to the printing options specified with this shell command. The printing options available are the same as those with the first form of the *lp* shell command. If the request has finished printing, the change is rejected. If the request is already printing, it will be stopped and restarted from the beginning (unless the **-P** option has been given).

lp associates a unique *request-id* with each request and prints it on the standard output. This *request-id* can be used later to *cancel*, *change*, or *find* the status of the request. (See the section on *cancel* for details about canceling a request, the previous paragraph for an explanation of how to change a request, and *lpstat*(1) for information about checking the status of a print request.)

Sending a Print Request

The first form of the *lp* command is used to send a print request to a particular printer or group of printers.

Options to *lp* must always precede file names, but may be listed in any order. The following options are available for *lp*:

- c** Make copies of the *files* to be printed immediately when *lp* is invoked. Normally, *files* will not be copied. If the **-c** option is not given, then the user should be careful not to remove any of the *files* before the request has been printed in its entirety. It should also be noted that in the absence of the **-c** option, any changes made to the named *files* after the request is made but before it is printed will be reflected in the printed output.
- d** *dest* Print this request using *dest* as the printer or class of printers. Under certain conditions (lack of printer availability, capabilities of printers, and so on), requests for specific destinations may not be accepted (see *accept*(1M) and *lpstat*(1)). By default, *dest* is taken from the environment variable LPDEST (if it is set). Otherwise, a default destination (if one exists) for the computer system is used. Destination names vary from system to system (see *lpstat*(1)).

-f *form-name* [**-d any**]

Print the request on the form *form-name*. The LP print service ensures that the form is mounted on the printer. If *form-name* is requested with a printer destination that cannot support the form, the request is rejected. If *form-name* has not been defined for the system, or if the user is not allowed to use the form, the request is rejected (see *lpforms(1M)*). When the **-d any** option is given, the request is printed on any printer that has the requested form mounted and can handle any other needs of the print request.

-H *special-handling*

Print the request according to the value of *special-handling*. Acceptable values for *special-handling* are **hold**, **resume**, and **immediate**, as defined below:

hold Don't print the request until notified. If printing has already begun, stop it. Other print requests will go ahead of a held request until it is resumed.

resume Resume a held request. If it had been printing when held, it will be the next request printed, unless subsequently bumped by an **immediate** request.

immediate

(Available only to LP administrators)

Print the request next. If more than one request is assigned **immediate**, the requests are printed in the reverse order queued. If a request is currently printing on the desired printer, you have to put it on hold to allow the immediate request to print.

-m Send mail (see *mail(1)*) after the files have been printed. By default, no mail is sent upon normal completion of the print request.

-n *number* Print *number* copies of the output. (Default is 1.)

-o *option* Specify printer-dependent or class-dependent *options*. Several such options may be specified on a single command line either by using the **-o** keyletter more than once (i.e., **-o option₁ -o option₂ ... -o option_n**), or by specifying a list of options with more than one **-o** keyletter (i.e., **-o option₁, option₂, ... option_n**). The standard interface recognizes the following options:

nobanner

Do not print a banner page with this request. (The administrator can disallow this option at any time.)

nofilebreak

Do not insert a form feed between the files given, if submitting a job to print more than one file.

length=*scaled-decimal-number*

Print this request with pages *scaled-decimal-number* lines long. A *scaled-decimal-number* is an optionally scaled decimal number

that gives a size in lines, columns, inches, or centimeters, as appropriate. The scale is indicated by appending the letter "i" for inches, or the letter "c" for centimeters. For length or width settings, an unscaled number indicates lines or columns; for line pitch or character pitch settings, an unscaled number indicates lines per inch or characters per inch (the same as a number scaled with "i"). For example, **length=66** indicates a page length of 66 lines, **length=11i** indicates a page length of 11 inches, and **length=27.94c** indicates a page length of 27.94 centimeters.

This option cannot be used with the **-f** option.

width=scaled-decimal-number

Print this request with page-width set to *scaled-decimal-number* columns wide. (See the explanation of *scaled-decimal-numbers* in the discussion of **length**, above.) This option cannot be used with the **-f** option.

lpi=scaled-decimal-number

Print this request with the line pitch set to *scaled-decimal-number* lines per inch. This option cannot be used with the **-f** option.

cpi=scaled-decimal-number

Print this request with the character pitch set to *scaled-decimal-number* characters per inch. Character pitch can also be set to **pica** (representing 10 columns per inch) or **elite** (representing 12 columns per inch), or it can be **compressed** (representing as many columns as a printer can handle). There is no standard number of columns per inch for all printers; see the Terminfo database (*terminfo(4)*) for the default character pitch for your printer. This option cannot be used with the **-f** option.

stty=stty-option-list

A list of options valid for the **stty** command; enclose the list with quotes if it contains blanks.

-P page-list Print the pages specified in *page-list*. This option can be used only if there is a filter available to handle it; otherwise, the print request will be rejected. The *page-list* may consist of range(s) of numbers, single page numbers, or a combination of both. The pages will be printed in ascending order.

-q priority-level

Assign this request *priority-level* in the printing queue. The values of *priority-level* range from 0, the highest priority, to 39, the lowest priority. If a priority is not specified, the default for the print service is used, as assigned by the system administrator.

-s

Suppress messages from the print service such as `request id is request-id`.

-S *character-set* [**-d** *any*]

-S *print-wheel* [**-d** *any*]

Print this request using the specified *character-set* or *print-wheel*. If a form was requested and it requires a character set or print wheel other than the one specified with the **-S** option, the request is rejected. For printers that take print wheels: if the print wheel specified is not one listed by the administrator as acceptable for the printer specified in this request, the request is rejected unless the print wheel is already mounted on the printer. For printers that use selectable or programmable character sets: if the *character-set* specified is not one defined in the Terminfo database for the printer (see *terminfo(4)*), or is not an alias defined by the administrator, the request is rejected. When the **-d** *any* option is used, the request is printed on any printer that has the print wheel mounted or any printer that can select the character set, and that can handle any other needs of the request.

-t *title* Print *title* on the banner page of the output. The default is no title.

-T *content-type* [**-r**]

Print the request on a printer that can support the specified *content-type*. If no printer accepts this type directly, a filter will be used to convert the content into an acceptable type. If the **-r** option is specified, a filter will not be used. If **-r** is specified, but no printer accepts the *content-type* directly, the request is rejected. If the *content-type* is not acceptable to any printer, either directly or with a filter, the request is rejected.

-w Write a message on the user's terminal after the *files* have been printed. If the user is not logged in, then mail will be sent instead.

-y *mode-list*

Print this request according to the printing modes listed in *mode-list*. The allowed values for *mode-list* are locally defined. This option can be used only if there is a filter available to handle it; otherwise, the print request will be rejected.

Canceling a Print Request

The *cancel* command cancels printer requests that were made by the *lp(1)* shell command. The shell command line arguments may be either *request-ids* (as returned by *lp(1)*) or *printer* names (for a complete list, use *lpstat(1)*). Specifying a *request-id* cancels the associated request even if it is currently printing. Specifying a *printer* cancels the request that is currently printing on that printer. In either case, the cancellation of a request that is currently printing frees the printer to print its next available request.

NOTES

Printers for which requests are not being accepted will not be considered when the destination is *any*. (Use the *lpstat -a* command to see which printers are accepting requests.) On the other hand, if a request is destined

for a class of printers and the class itself is accepting requests, *all* printers in the class will be considered, regardless of their acceptance status, as long as the printer class is accepting requests.

WARNING

For printers that take mountable print wheels or font cartridges, if you do not specify a particular print wheel or font with the `-S` option, whichever one happens to be mounted at the time your request is printed will be used. Use the `lpstat -p printer -l` command to see which print wheels are available on a particular printer, or the `lpstat -S -l` command to find out what print wheels are available and on which printers. For printers that have selectable character sets, you will get the standard character set if you don't use the `-S` option.

FILES

`/usr/spool/lp/*`

SEE ALSO

`enable(1)`, `lpstat(1)`, `mail(1)`.
`accept(1M)`, `lpadmin(1M)`, `lpfilter(1M)`, `lpforms(1M)`, `lpsched(1M)`, `lpusers(1M)`
in the *System Administrator's Reference Manual*.
`terminfo(4)` in the *Programmer's Reference Manual*.



NAME

`lpstat` – print information about the status of the LP print service

SYNOPSIS

`lpstat` [options]

DESCRIPTION

`lpstat` prints information about the current status of the LP print service.

If no options are given, then `lpstat` prints the status of all requests made to `lp(1)` by users. Any arguments that are not options are assumed to be *request-ids* (as returned by `lp`), printers, or printer classes. `lpstat` prints the status of such requests, printers, or printer classes. Options may appear in any order and may be repeated and intermixed with other arguments. Some of the keyletters below may be followed by an optional *list* that can be in one of two forms: a list of items separated from one another by a comma, or a list of items enclosed in double quotes and separated from one another by a comma and/or one or more spaces. For example:

`-u "user1, user2, user3"`

Specifying "all" after any keyletters that take "list" as an argument causes all information relevant to the keyletter to be printed. For example, the command

`lpstat -o all`

prints the status of all output requests.

- `-a [list]` Display acceptance status (with respect to `lp`) of destinations for requests (see `accept(1M)`). *List* is a list of intermixed printer names and class names; the default is **all**.
- `-c [list]` Display class names and their members. *List* is a list of class names; the default is **all**.
- `-d` Display the system default destination for `lp`.
- `-f [list] [-l]` Display a verification that the forms in *list* are recognized by the LP print service. *List* is a list of forms; the default is **all**. The `-l` option will list the form descriptions.
- `-o [list] [-l]` Display the status of output requests. *List* is a list of intermixed printer names, class names, and request-ids; the default is **all**. The `-l` option gives a more detailed status of the request.
- `-p [list] [-D] [-l]` Display the status of printers named in *list*. *List* is a list of printers; the default is **all**. If the `-D` option is given, a brief description is printed for each printer in *list*. If the `-l` option is given, a full description of each printer's configuration is given, including the form mounted, the acceptable content and printer types, a printer description, the interface used, and so on.
- `-r` Display the status of the LP request scheduler.

- s Display a status summary, including the system default destination, a list of class names and their members, a list of printers and their associated devices, a list of all forms currently mounted, and a list of all recognized character sets and print wheels.
- S [*list*] [-1] Display a verification that the character sets or the print wheels specified in *list* are recognized by the LP print service. Items in *list* can be character sets or print wheels; the default for the list is **all**. If the -1 option is given, each line is appended by a list of printers that can handle the print wheel or character set. The list also shows whether the print wheel or character set is mounted or specifies the built-in character set into which it maps.
- t Display all status information.
- u [*list*] Display the status of output requests for users. *List* is a list of login names; the default is **all**.
- v [*list*] Display the names of printers and the path names of the devices associated with them. *List* is a list of printer names; the default is **all**.

FILES

/usr/spool/lp/*

SEE ALSO

enable(1), lp(1).

NAME

`ls` – list contents of directory

SYNOPSIS

`ls [-RadCxmlnogrtucpFbqisf] [names]`

DESCRIPTION

For each directory argument, *ls* lists the contents of the directory; for each file argument, *ls* repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The default format is to list one entry per line, the `-C` and `-x` options enable multi-column formats, and the `-m` option enables stream output format. In order to determine output formats for the `-C`, `-x`, and `-m` options, *ls* uses an environment variable, `COLUMNS`, to determine the number of character positions available on one output line. If this variable is not set, the *terminfo*(4) database is used to determine the number of columns, based on the environment variable `TERM`. If this information cannot be obtained, 80 columns are assumed.

The *ls* command has the following options:

- `-R` Recursively list subdirectories encountered.
- `-a` List all entries, including those that begin with a dot (`.`), which are normally not listed.
- `-d` If an argument is a directory, list only its name (not its contents); often used with `-l` to get the status of a directory.
- `-C` Multi-column output with entries sorted down the columns.
- `-x` Multi-column output with entries sorted across rather than down the page.
- `-m` Stream output format; files are listed across the page, separated by commas.
- `-l` List in long format, giving mode, number of links, owner, group, size in bytes, and time of last modification for each file (see below). If the file is a special file, the size field will instead contain the major and minor device numbers rather than a size.
- `-n` The same as `-l`, except that the owner's `UID` and group's `GID` numbers are printed, rather than the associated character strings.
- `-o` The same as `-l`, except that the group is not printed.
- `-g` The same as `-l`, except that the owner is not printed.
- `-r` Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.
- `-t` Sort by time stamp (latest first) instead of by name. The default is the last modification time. (See `-n` and `-c`.)

- u Use time of last access instead of last modification for sorting (with the -t option) or printing (with the -l option).
- c Use time of last modification of the i-node (file created, mode changed, etc.) for sorting (-t) or printing (-l).
- p Put a slash (/) after each filename if that file is a directory.
- F Put a slash (/) after each filename if that file is a directory and put an asterisk (*) after each filename if that file is executable.
- b Force printing of non-printable characters to be in the octal \ddd notation.
- q Force printing of non-printable characters in file names as the character question mark (?).
- i For each file, print the i-number in the first column of the report.
- s Give size in blocks, including indirect blocks, for each entry.
- f Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory.

The mode printed under the -l option consists of ten characters. The first character may be one of the following:

- d the entry is a directory;
- b the entry is a block special file;
- c the entry is a character special file;
- p the entry is a fifo (a.k.a. "named pipe") special file;
- the entry is an ordinary file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the file; and the last to all others. Within each set, the three characters indicate permission to read, to write, and to execute the file as a program, respectively. For a directory, "execute" permission is interpreted to mean permission to search the directory for a specified file.

ls -l (the long list) prints its output as follows:

```
-rwxrwxrwx 1 smith dev 10876 May 16 9:42 part2
```

This horizontal configuration provides a good deal of information. Reading from right to left, you see that the current directory holds one file, named "part2." Next, the last time that file's contents were modified was 9:42 A.M. on May 16. The file is moderately sized, containing 10,876 characters, or bytes. The owner of the file, or the user, belongs to the group "dev" (perhaps indicating "development"), and his or her login name is "smith." The number, in this case "1," indicates the number of links to file "part2." Finally, the row of dash and letters tell you that user, group, and others have permissions to read, write, execute "part2."

The execute (x) symbol here occupies the third position of the three-character sequence. A - in the third position would have indicated a denial of execution permissions.

The permissions are indicated as follows:

- r the file is readable
- w the file is writable
- x the file is executable
- the indicated permission is *not* granted
- l mandatory locking will occur during access (the set-group-ID bit is on and the group execution bit is off)
- s the set-user-ID or set-group-ID bit is on, and the corresponding user or group execution bit is also on
- S undefined bit-state (the set-user-ID bit is on and the user execution bit is off)
- t the 1000 (octal) bit, or sticky bit, is on (see *chmod(1)*), and execution is on
- T the 1000 bit is turned on, and execution is off (undefined bit-state)

For user and group permissions, the third position is sometimes occupied by a character other than x or -. s also may occupy this position, referring to the state of the set-ID bit, whether it be the user's or the group's. The ability to assume the same ID as the user during execution is, for example, used during login when you begin as root but need to assume the identity of the user stated at "login."

In the case of the sequence of group permissions, l may occupy the third position. l refers to mandatory file and record locking. This permission describes a file's ability to allow other files to lock its reading or writing permissions during access.

For others permissions, the third position may be occupied by t or T. These refer to the state of the sticky bit and execution permissions.

EXAMPLES

An example of a file's permissions is:

```
-rwxr--r--
```

This describes a file that is readable, writable, and executable by the user and readable by the group and others.

Another example of a file's permissions is:

```
-rwsr-xr-x
```

This describes a file that is readable, writable, and executable by the user, readable and executable by the group and others, and allows its user-ID to be assumed, during execution, by the user presently executing it.

Another example of a file's permissions is:

```
-rw-rwl---
```

This describes a file that is readable and writable only by the user and the group and can be locked during access.

An example of a command line:

```
ls -a
```

This command will print the names of all files in the current directory, including those that begin with a dot (.), which normally do not print.

Another example of a command line:

```
ls -aisn
```

This command will provide you with quite a bit of information including all files, including non-printing ones (a), the i-number—the memory address of the i-node associated with the file—printed in the left-hand column (i); the size (in blocks) of the files, printed in the column to the right of the i-numbers (s); finally, the report is displayed in the numeric version of the long list, printing the UID (instead of user name) and GID (instead of group name) numbers associated with the files.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

FILES

/etc/passwd	user IDs for <code>ls -l</code> and <code>ls -o</code>
/etc/group	group IDs for <code>ls -l</code> and <code>ls -g</code>
/usr/lib/terminfo/?/*	terminal information database

SEE ALSO

`chmod(1)`, `find(1)`.

NOTES

In a Remote File Sharing environment, you may not have the permissions that the output of the `ls -l` command leads you to believe. For more information see the "Mapping Remote Users" section of Chapter 10 of the *System Administrator's Guide*.

BUGS

Unprintable characters in file names may confuse the columnar output options.

NAME

machid: pdp11, u3b, u3b2, u3b5, vax – get processor type truth value

SYNOPSIS

pdp11

u3b

u3b2

u3b5

vax

DESCRIPTION

The following commands will return a true value (exit code of 0) if you are on a processor that the command name indicates.

pdp11 True if you are on a PDP-11/45 or PDP-11/70.

u3b True if you are on a 3B20 computer.

u3b2 True if you are on a 3B2 computer.

u3b5 True if you are on a 3B5 computer.

vax True if you are on a VAX-11/750 or VAX-11/780.

The commands that do not apply will return a false (non-zero) value. These commands are often used within makefiles (see *make(1)*) and shell procedures (see *sh(1)*) to increase portability.

SEE ALSO

sh(1), *test(1)*, *true(1)*.

make(1) in the *Programmer's Reference Manual*.



NAME

`mail`, `rmail` – send mail to users or read mail

SYNOPSIS

Sending mail:

`mail` [`-wt`] persons

`rmail` [`-wt`] persons

Reading mail:

`mail` [`-ehpqr`] [`-f file`] [`-F persons`]

DESCRIPTION

Sending mail:

The command-line options that follow affect SENDING mail:

- `-w` causes a letter to be sent to a remote user without waiting for the completion of the remote transfer program.
- `-t` causes a **To:** line to be added to the letter, showing the intended recipients.

A *person* is usually a user name recognized by `login(1)`. When *persons* are named, `mail` assumes a message is being sent (except in the case of the `-F` option). It reads from the standard input up to an end-of-file (control-d), or until it reads a line consisting of just a period. When either of those signals is received, `mail` adds the *letter* to the *mailfile* for each *person*. A *letter* is a message preceded by a *postmark*. The message is preceded by the sender's name and a *postmark*. A *postmark* consists of one or more 'From' lines followed by a blank line.

If a letter is found to be undeliverable, it is returned to the sender with diagnostics that indicate the location and nature of the failure. If `mail` is interrupted during input, the file `dead.letter` is saved to allow editing and resending. `dead.letter` is recreated every time it is needed, erasing any previous contents.

`rmail` only permits the sending of mail; `uucp(1C)` uses `rmail` as a security precaution.

If the local system has the Basic Networking Utilities installed, mail may be sent to a recipient on a remote system. Prefix *person* by the system name and exclamation point. A series of system names separated by exclamation points can be used to direct a letter through an extended network.

Reading Mail:

The command-line options that follow affect READING mail:

- `-e` causes mail not to be printed. An exit value of 0 is returned if the user has mail; otherwise, an exit value of 1 is returned.
- `-h` causes a window of headers to be displayed rather than the latest message. The display is followed by the '?' prompt.
- `-p` causes all messages to be printed without prompting for disposition.
- `-q` causes `mail` to terminate after interrupts. Normally an interrupt causes only the termination of the message being printed.

- r causes messages to be printed in first-in, first-out order.
- f*file* causes *mail* to use *file* (e.g., **mbox**) instead of the default *mailfile*.
- F*persons*
entered into an empty *mailbox*, causes all incoming mail to be forwarded to *persons*.

mail, unless otherwise influenced by command-line options, prints a user's mail messages in last-in, first-out order. For each message, the user is prompted with a *?*, and a line is read from the standard input. The following commands are available to determine the disposition of the message:

- <new-line>, +, or n Go on to next message.
- d, or dp Delete message and go on to next message.
- d # Delete message number #. Do not go on to next message.
- dq Delete message and quit *mail*.
- h Display a window of headers around current message.
- h # Display header of message number #.
- h a Display headers of ALL messages in the user's *mailfile*.
- h d Display headers of messages scheduled for deletion.
- p Print current message again.
- Print previous message.
- a Print message that arrived during the *mail* session.
- # Print message number #.
- r [*users*] Reply to the sender, and other *user(s)*, then delete the message.
- s [*files*] Save message in the named *files* (**mbox** is default).
- y Same as save.
- u [#] Undelete message number # (default is last read).
- w [*files*] Save message, without its top-most header, in the named *files* (**mbox** is default).
- m [*persons*] Mail the message to the named *persons*.
- q, or ctl-d Put undeleted mail back in the *mailfile* and quit *mail*.
- x Put all mail back in the *mailfile* unchanged and exit *mail*.
- !*command* Escape to the shell to do *command*.
- ? Print a command summary.

When a user logs in, the presence of mail, if any, is indicated. Also, notification is made if new mail arrives while using *mail*.

The *mailfile* may be manipulated in two ways to alter the function of *mail*. The *other* permissions of the file may be read-write, read-only, or neither read

nor write to allow different levels of privacy. If changed to other than the default, the file will be preserved even when empty to perpetuate the desired permissions. The file may also contain the first line:

Forward to *person*

which will cause all mail sent to the owner of the *mailfile* to be forwarded to *person*. A "Forwarded by..." message is added to the header. This is especially useful in a multi-machine environment to forward all of a person's mail to a single machine, and to keep the recipient informed if the mail has been forwarded. Installation and removal of forwarding is done with the `-F` option.

To forward all of one's mail to `systema!user` enter:

`mail -Fsystema!user`

To forward to more than one user enter:

`mail -F"user1,systema!user2,systema!systemb!user3"`

Note that when more than one user is specified, the whole list should be enclosed in double quotes so that it may all be interpreted as the operand of the `-F` option. The list can be up to 1024 bytes; either commas or white space can be used to separate users.

The following list of characters are prohibited from appearing anywhere in the mail `-F` argument list or in the "Forward to" line:

; & | ^ < > ` () <CR>

To remove forwarding enter:

`mail -F ""`

The pair of double quotes is mandatory to set a NULL argument for the `-F` option.

In order for forwarding to work properly the *mailfile* should have "mail" as group ID, and the group permission should be read-write.

FILES

<code>/etc/passwd</code>	to identify sender and locate persons
<code>/usr/mail/user</code>	incoming mail for <i>user</i> ; i.e., the <i>mailfile</i>
<code>\$HOME/mbox</code>	saved mail
<code>\$MAIL</code>	variable containing path name of <i>mailfile</i>
<code>/tmp/ma*</code>	temporary file
<code>/usr/mail/*.lock</code>	lock for mail directory
<code>dead.letter</code>	unmailable text

SEE ALSO

`login(1)`, `mailx(1)`, `write(1)`.
User's Guide.
System Administrator's Guide.

WARNINGS

The "Forward to person" feature may result in a loop, if *sys1!userb* forwards to *sys2!userb* and *sys2!userb* forwards to *sys1!userb*. The symptom is a message saying "unbounded...saved mail in dead.letter."

If 3bnet is to be used to send mail, `/usr/3bnet/lib/3bnet` must be executable to others.

BUGS

Conditions sometimes result in a failure to remove a lock file.

After an interrupt, the next message may not be printed; printing may be forced by typing a p.

NAME

mailx – interactive message processing system

SYNOPSIS

mailx [*options*] [*name...*]

DESCRIPTION

The command *mailx* provides a comfortable, flexible environment for sending and receiving messages electronically. When reading mail, *mailx* provides commands to facilitate saving, deleting, and responding to messages. When sending mail, *mailx* allows editing, reviewing and other modification of the message as it is entered.

Many of the remote features of *mailx* will only work if the Basic Networking Utilities are installed on your system.

Incoming mail is stored in a standard file for each user, called the *mailbox* for that user. When *mailx* is called to read messages, the *mailbox* is the default place to find them. As messages are read, they are marked to be moved to a secondary file for storage, unless specific action is taken, so that the messages need not be seen again. This secondary file is called the *mbox* and is normally located in the user's HOME directory (see "MBOX" (ENVIRONMENT VARIABLES) for a description of this file). Messages can be saved in other secondary files named by the user. Messages remain in a secondary file until forcibly removed.

The user can access a secondary file by using the `-f` option of the *mailx* command. Messages in the secondary file can then be read or otherwise processed using the same COMMANDS as in the primary *mailbox*. This gives rise within these pages to the notion of a current *mailbox*.

On the command line, *options* start with a dash (`-`) and any other arguments are taken to be destinations (recipients). If no recipients are specified, *mailx* will attempt to read messages from the *mailbox*. Command line options are:

- `-e` Test for presence of mail. *mailx* prints nothing and exits with a successful return code if there is mail to read.
- `-f [filename]` Read messages from *filename* instead of *mailbox*. If no *filename* is specified, the *mbox* is used.
- `-F` Record the message in a file named after the first recipient. Overrides the "record" variable, if set (see ENVIRONMENT VARIABLES).
- `-h number` The number of network "hops" made so far. This is provided for network software to avoid infinite delivery loops. (See `addsopt` under ENVIRONMENT VARIABLES)
- `-H` Print header summary only.
- `-i` Ignore interrupts. See also "ignore" (ENVIRONMENT VARIABLES).
- `-n` Do not initialize from the system default *mailx.rc* file.

- N** Do not print initial header summary.
- r *address*** Pass *address* to network delivery software. All tilde commands are disabled. (See **addsopt** under ENVIRONMENT VARIABLES)
- s *subject*** Set the Subject header field to *subject*.
- u *user*** Read *user's mailbox*. This is only effective if *user's mailbox* is not read protected.
- U** Convert *uucp* style addresses to internet standards. Overrides the "conv" environment variable. (See **addsopt** under ENVIRONMENT VARIABLES)

When reading mail, *mailx* is in *command mode*. A header summary of the first several messages is displayed, followed by a prompt indicating *mailx* can accept regular commands (see COMMANDS below). When sending mail, *mailx* is in *input mode*. If no subject is specified on the command line, a prompt for the subject is printed. (A "subject" longer than 1024 characters will cause *mailx* to dump core) As the message is typed, *mailx* will read the message and store it in a temporary file. Commands may be entered by beginning a line with the tilde (~) escape character followed by a single command letter and optional arguments. See TILDE ESCAPES for a summary of these commands.

At any time, the behavior of *mailx* is governed by a set of *environment variables*. These are flags and valued parameters which are set and cleared via the **set** and **unset** commands. See ENVIRONMENT VARIABLES below for a summary of these parameters.

Recipients listed on the command line may be of three types: login names, shell commands, or alias groups. Login names may be any network address, including mixed network addressing. If mail is found to be undeliverable, an attempt is made to return it to the sender's *mailbox*. If the recipient name begins with a pipe symbol (|), the rest of the name is taken to be a shell command to pipe the message through. This provides an automatic interface with any program that reads the standard input, such as *lp(1)* for recording outgoing mail on paper. Alias groups are set by the **alias** command (see COMMANDS below) and are lists of recipients of any type.

Regular commands are of the form

```
[ command ] [ msglist ] [ arguments ]
```

If no command is specified in *command mode*, **print** is assumed. In *input mode*, commands are recognized by the escape character, and lines not treated as commands are taken as input for the message.

Each message is assigned a sequential number, and there is at any time the notion of a current message, marked by a right angle bracket (>) in the header summary. Many commands take an optional list of messages (*msglist*) to operate on. The default for *msglist* is the current message. A *msglist* is a list of message identifiers separated by spaces, which may include:

```
n      Message number n.
```

. The current message.
 ^ The first undeleted message.
 \$ The last message.
 * All messages.
n-m An inclusive range of message numbers.
user All messages from **user**.
/string All messages with **string** in the subject line (case ignored).
:c All messages of type *c*, where *c* is one of:
 d deleted messages
 n new messages
 o old messages
 r read messages
 u unread messages

Note that the context of the command determines whether this type of message specification makes sense.

Other arguments are usually arbitrary strings whose usage depends on the command involved. File names, where expected, are expanded via the normal shell conventions (see *sh(1)*). Special characters are recognized by certain commands and are documented with the commands below.

At start-up time, *mailx* tries to execute commands from the optional system-wide file (`/usr/lib/mailx/mailx.rc`) to initialize certain parameters, then from a private start-up file (`$HOME/.mailrc`) for personalized variables. With the exceptions noted below, regular commands are legal inside start-up files. The most common use of a start-up file is to set up initial display options and alias lists. The following commands are not legal in the start-up file: **!**, **Copy**, **edit**, **followup**, **Followup**, **hold**, **mail**, **preserve**, **reply**, **Reply**, **shell**, and **visual**. An error in the start-up file causes the remaining lines in the file to be ignored. The `.mailrc` file is optional, and must be constructed locally.

COMMANDS

The following is a complete list of *mailx* commands:

!shell-command

Escape to the shell. See "SHELL" (ENVIRONMENT VARIABLES).

comment

Null command (comment). This may be useful in `.mailrc` files.

-

Print the current message number.

?

Prints a summary of commands.

alias alias name ...

group alias name ...

Declare an alias for the given names. The names will be substituted when *alias* is used as a recipient. Useful in the `.mailrc` file.

alternates *name* ...

Declares a list of alternate names for your login. When responding to a message, these names are removed from the list of recipients for the response. With no arguments, **alternates** prints the current list of alternate names. See also "allnet" (ENVIRONMENT VARIABLES).

cd [*directory*]

chdir [*directory*]

Change directory. If *directory* is not specified, \$HOME is used.

copy [*filename*]

copy [*msglist*] *filename*

Copy messages to the file without marking the messages as saved. Otherwise equivalent to the save command.

Copy [*msglist*]

Save the specified messages in a file whose name is derived from the author of the message to be saved, without marking the messages as saved. Otherwise equivalent to the Save command.

delete [*msglist*]

Delete messages from the *mailbox*. If "autoprint" is set, the next message after the last one deleted is printed (see ENVIRONMENT VARIABLES).

discard [*header-field* ...]

ignore [*header-field* ...]

Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are "status" and "cc." The fields are included when the message is saved. The Print and Type commands override this command.

dp [*msglist*]

dt [*msglist*]

Delete the specified messages from the *mailbox* and print the next message after the last one deleted. Roughly equivalent to a delete command followed by a print command.

echo *string* ...

Echo the given strings (like *echo*(1)).

edit [*msglist*]

Edit the given messages. The messages are placed in a temporary file and the "EDITOR" variable is used to get the name of the editor (see ENVIRONMENT VARIABLES). Default editor is *ed*(1).

exit
xit

Exit from *mailx*, without changing the *mailbox*. No messages are saved in the *mbox* (see also **quit**).

file [*filename*]

folder [*filename*]

Quit from the current file of messages and read in the specified file. Several special characters are recognized when used as file names, with the following substitutions:

% the current *mailbox*.

%*user* the *mailbox* for *user*.

the previous file.

& the current *mbox*.

Default file is the current *mailbox*.

folders

Print the names of the files in the directory set by the "folder" variable (see ENVIRONMENT VARIABLES).

followup [*message*]

Respond to a message, recording the response in a file whose name is derived from the author of the message. Overrides the "record" variable, if set. See also the Followup, Save, and Copy commands and "outfolder" (ENVIRONMENT VARIABLES).

Followup [*msglist*]

Respond to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line is taken from the first message and the response is recorded in a file whose name is derived from the author of the first message. See also the followup, Save, and Copy commands and "outfolder" (ENVIRONMENT VARIABLES).

from [*msglist*]

Prints the header summary for the specified messages.

group *alias name* ...

alias *alias name* ...

Declare an alias for the given names. The names will be substituted when *alias* is used as a recipient. Useful in the *.mailrc* file.

headers [*message*]

Prints the page of headers which includes the message specified. The "screen" variable sets the number of headers per page (see ENVIRONMENT VARIABLES). See also the **z** command.

help

Prints a summary of commands.

hold [*msglist*]**preserve** [*msglist*]

Holds the specified messages in the *mailbox*.

if *s* | *r*

mail-commands

else

mail-commands

endif

Conditional execution, where *s* will execute following *mail-commands*, up to an **else** or **endif**, if the program is in *send* mode, and *r* causes the *mail-commands* to be executed only in *receive* mode. Useful in the *.mailrc* file.

ignore *header-field ...***discard** *header-field ...*

Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are "status" and "cc." All fields are included when the message is saved. The **Print** and **Type** commands override this command.

list

Prints all commands available. No explanation is given.

mail *name ...*

Mail a message to the specified users.

Mail *name*

Mail a message to the specified user and record a copy of it in a file named after that user.

mbox [*msglist*]

Arrange for the given messages to end up in the standard *mbox* save file when *mailx* terminates normally. See "MBOX" (ENVIRONMENT VARIABLES) for a description of this file. See also the **exit** and **quit** commands.

next [*message*]

Go to next message matching *message*. A *msglist* may be specified, but in this case the first valid message in the list is the only one used. This is useful for jumping to the next message from a specific user, since the name would be taken as a command in the absence of a real command. See the discussion of *msglists* above for a description of possible message specifications.

pipe [*msglist*] [*shell-command*]
[[*msglist*] [*shell-command*]

Pipe the message through the given *shell-command*. The message is treated as if it were read. If no arguments are given, the current message is piped through the command specified by the value of the "cmd" variable. If the "page" variable is set, a form feed character is inserted after each message (see ENVIRONMENT VARIABLES).

preserve [*msglist*]
hold [*msglist*]

Preserve the specified messages in the *mailbox*.

Print [*msglist*]
Type [*msglist*]

Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the *ignore* command.

print [*msglist*]
type [*msglist*]

Print the specified messages. If "crt" is set, the messages longer than the number of lines specified by the "crt" variable are paged through the command specified by the "PAGER" variable. The default command is *pg*(1) (see ENVIRONMENT VARIABLES).

quit

Exit from *mailx*, storing messages that were read in *mbox* and unread messages in the *mailbox*. Messages that have been explicitly saved in a file are deleted.

Reply [*msglist*]
Respond [*msglist*]

Send a response to the author of each message in the *msglist*. The subject line is taken from the first message. If "record" is set to a file name, the response is saved at the end of that file (see ENVIRONMENT VARIABLES).

reply [*message*]
respond [*message*]

Reply to the specified message, including all other recipients of the message. If "record" is set to a file name, the response is saved at the end of that file (see ENVIRONMENT VARIABLES).

Save [*msglist*]

Save the specified messages in a file whose name is derived from the author of the first message. The name of the file is taken to be the author's name with all network addressing stripped off. See also the *Copy*, *followup*, and *Followup* commands and "outfolder" (ENVIRONMENT VARIABLES).

save [*filename*]

save [*msglist*] *filename*

Save the specified messages in the given file. The file is created if it does not exist. The message is deleted from the *mailbox* when *mailx* terminates unless "keepsave" is set (see also ENVIRONMENT VARIABLES and the *exit* and *quit* commands).

set

set *name*

set *name*=*string*

set *name*=*number*

Define a variable called *name*. The variable may be given a null, string, or numeric value. Set by itself prints all defined variables and their values. See ENVIRONMENT VARIABLES for detailed descriptions of the *mailx* variables.

shell

Invoke an interactive shell (see also "SHELL" (ENVIRONMENT VARIABLES)).

size [*msglist*]

Print the size in characters of the specified messages.

source *filename*

Read commands from the given file and return to command mode.

top [*msglist*]

Print the top few lines of the specified messages. If the "toplines" variable is set, it is taken as the number of lines to print (see ENVIRONMENT VARIABLES). The default is 5.

touch [*msglist*]

Touch the specified messages. If any message in *msglist* is not specifically saved in a file, it will be placed in the *mbox*, or the file specified in the MBOX environment variable, upon normal termination. See *exit* and *quit*.

Type [*msglist*]

Print [*msglist*]

Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the *ignore* command.

type [*msglist*]

print [*msglist*]

Print the specified messages. If "crt" is set, the messages longer than the number of lines specified by the "crt" variable are paged through the command specified by the "PAGER" variable. The default command is *pg*(1) (see ENVIRONMENT VARIABLES).

undelete [*msglist*]

Restore the specified deleted messages. Will only restore messages deleted in the current mail session. If "autoprint" is set, the last message of those restored is printed (see ENVIRONMENT VARIABLES).

unset *name* ...

Causes the specified variables to be erased. If the variable was imported from the execution environment (i.e., a shell variable) then it cannot be erased.

version

Prints the current version and release date.

visual [*msglist*]

Edit the given messages with a screen editor. The messages are placed in a temporary file and the "VISUAL" variable is used to get the name of the editor (see ENVIRONMENT VARIABLES).

write [*msglist*] *filename*

Write the given messages on the specified file, minus the header and trailing blank line. Otherwise equivalent to the save command.

xit**exit**

Exit from *mailx*, without changing the *mailbox*. No messages are saved in the *mbox* (see also quit).

z[+|-]

Scroll the header display forward or backward one screen—full. The number of headers displayed is set by the "screen" variable (see ENVIRONMENT VARIABLES).

TILDE ESCAPES

The following commands may be entered only from *input mode*, by beginning a line with the tilde escape character (~). See "escape" (ENVIRONMENT VARIABLES) for changing this special character.

~! *shell-command*

Escape to the shell.

~.

Simulate end of file (terminate message input).

~: *mail-command***~-** *mail-command*

Perform the command-level request. Valid only when sending a message while reading mail.

- `~?` Print a summary of tilde escapes.
- `~A` Insert the autograph string "Sign" into the message (see ENVIRONMENT VARIABLES).
- `~a` Insert the autograph string "sign" into the message (see ENVIRONMENT VARIABLES).
- `~b name ...` Add the *names* to the blind carbon copy (Bcc) list.
- `~c name ...` Add the *names* to the carbon copy (Cc) list.
- `~d` Read in the *dead.letter* file. See "DEAD" (ENVIRONMENT VARIABLES) for a description of this file.
- `~e` Invoke the editor on the partial message. See also "EDITOR" (ENVIRONMENT VARIABLES).
- `~f [msglist]` Forward the specified messages. The messages are inserted into the message without alteration.
- `~h` Prompt for Subject line and To, Cc, and Bcc lists. If the field is displayed with an initial value, it may be edited as if you had just typed it.
- `~i string` Insert the value of the named variable into the text of the message. For example, `~A` is equivalent to `'~i Sign.'` Environment variables set and exported in the shell are also accessible by `~i`.
- `~m [msglist]` Insert the specified messages into the letter, shifting the new text to the right one tab stop. Valid only when sending a message while reading mail.
- `~p` Print the message being entered.

~q

Quit from input mode by simulating an interrupt. If the body of the message is not null, the partial message is saved in *dead.letter*. See "DEAD" (ENVIRONMENT VARIABLES) for a description of this file.

~r filename**--< filename****---< !shell-command**

Read in the specified file. If the argument begins with an exclamation point (!), the rest of the string is taken as an arbitrary shell command and is executed, with the standard output inserted into the message.

~s string ...

Set the subject line to *string*.

~t name ...

Add the given *names* to the To list.

~v

Invoke a preferred screen editor on the partial message. See also "VISUAL" (ENVIRONMENT VARIABLES).

~w filename

Write the partial message onto the given file, without the header.

~x

Exit as with **~q** except the message is not saved in *dead.letter*.

~|shell-command

Pipe the body of the message through the given *shell-command*. If the *shell-command* returns a successful exit status, the output of the command replaces the message.

ENVIRONMENT VARIABLES

The following are environment variables taken from the execution environment and are not alterable within *mailx*.

HOME=directory

The user's base of operations.

MAILRC=filename

The name of the start-up file. Default is \$HOME/.mailrc.

The following variables are internal *mailx* variables. They may be imported from the execution environment or set via the **set** command at any time. The **unset** command may be used to erase variables.

addsopt

Enabled by default. If */bin/mail* is not being used as the deliverer, **noaddsopt** should be specified. (See WARNINGS below)

allnet

All network names whose last component (login name) match are treated as identical. This causes the *msglist* message specifications to behave similarly. Default is **noallnet**. See also the **alternates** command and the "metoo" variable.

append

Upon termination, append messages to the end of the *mbox* file instead of prepending them. Default is **noappend**.

askcc

Prompt for the Cc list after message is entered. Default is **noaskcc**.

asksub

Prompt for subject if it is not specified on the command line with the **-s** option. Enabled by default.

autoprint

Enable automatic printing of messages after **delete** and **undelete** commands. Default is **noautoprint**.

bang

Enable the special-casing of exclamation points (!) in shell escape command lines as in *vi(1)*. Default is **nobang**.

cmd=shell-command

Set the default command for the **pipe** command. No default value.

conv=conversion

Convert uucp addresses to the specified address style. The only valid conversion now is *internet*, which requires a mail delivery program conforming to the RFC822 standard for electronic mail addressing. Conversion is disabled by default. See also "sendmail" and the **-U** command line option.

crt=number

Pipe messages having more than *number* lines through the command specified by the value of the "PAGER" variable (*pg(1)* by default). Disabled by default.

DEAD=filename

The name of the file in which to save partial letters in case of untimely interrupt. Default is *\$HOME/dead.letter*.

debug

Enable verbose diagnostics for debugging. Messages are not delivered. Default is **nobug**.

dot

Take a period on a line by itself during input from a terminal as end-of-file. Default is **nodot**.

EDITOR=shell-command

The command to run when the `edit` or `~e` command is used. Default is `ed(1)`.

escape=c

Substitute `c` for the `~` escape character. Takes effect with next message sent.

folder=directory

The directory for saving standard mail files. User-specified file names beginning with a plus (+) are expanded by preceding the file name with this directory name to obtain the real file name. If `directory` does not start with a slash (/), `$HOME` is prepended to it. In order to use the plus (+) construct on a `mailx` command line, "folder" must be an exported `sh` environment variable. There is no default for the "folder" variable. See also "outfolder" below.

header

Enable printing of the header summary when entering `mailx`. Enabled by default.

hold

Preserve all messages that are read in the `mailbox` instead of putting them in the standard `mbox` save file. Default is **nohold**.

ignore

Ignore interrupts while entering messages. Handy for noisy dial-up lines. Default is **noignore**.

ignoreeof

Ignore end-of-file during message input. Input must be terminated by a period (.) on a line by itself or by the `~.` command. Default is **noignoreeof**. See also "dot" above.

keep

When the `mailbox` is empty, truncate it to zero length instead of removing it. Disabled by default.

keepsave

Keep messages that have been saved in other files in the `mailbox` instead of deleting them. Default is **nokeepsave**.

MBOX=filename

The name of the file to save messages which have been read. The `xit` command overrides this function, as does saving the message explicitly in another file. Default is `$HOME/mbx`.

metoo

If your login appears as a recipient, do not delete it from the list. Default is **no`metoo`**.

LISTER=shell-command

The command (and options) to use when listing the contents of the "folder" directory. The default is `ls(1)`.

onehop

When responding to a message that was originally sent to several recipients, the other recipient addresses are normally forced to be relative to the originating author's machine for the response. This flag disables alteration of the recipients' addresses, improving efficiency in a network where all machines can send directly to all other machines (i.e., one hop away).

outfolder

Causes the files used to record outgoing messages to be located in the directory specified by the "folder" variable unless the path name is absolute. Default is **no`outfolder`**. See "folder" above and the `Save`, `Copy`, `followup`, and `Followup` commands.

page

Used with the `pipe` command to insert a form feed after each message sent through the pipe. Default is **no`page`**.

PAGER=shell-command

The command to use as a filter for paginating output. This can also be used to specify the options to be used. Default is `pg(1)`.

prompt=string

Set the *command mode* prompt to *string*. Default is `"? "`.

quiet

Refrain from printing the opening message and version when entering *mailx*. Default is **no`quiet`**.

record=filename

Record all outgoing mail in *filename*. Disabled by default. See also "outfolder" above.

save

Enable saving of messages in *dead.letter* on interrupt or delivery error. See "DEAD" for a description of this file. Enabled by default.

screen=*number*

Sets the number of lines in a screen—full of headers for the **headers** command.

sendmail=*shell-command*

Alternate command for delivering messages. Default is */bin/rmail(1)*.

sendwait

Wait for background mailer to finish before returning. Default is **nosendwait**.

SHELL=*shell-command*

The name of a preferred command interpreter. Default is *sh(1)*.

showto

When displaying the header summary and the message is from you, print the recipient's name instead of the author's name.

sign=*string*

The variable inserted into the text of a message when the **~a** (autograph) command is given. No default (see also **~i** (TILDE ESCAPES)).

Sign=*string*

The variable inserted into the text of a message when the **~A** command is given. No default (see also **~i** (TILDE ESCAPES)).

toplines=*number*

The number of lines of header to print with the **top** command. Default is 5.

VISUAL=*shell-command*

The name of a preferred screen editor. Default is *vi(1)*.

FILES

<i>\$HOME/.mailrc</i>	personal start-up file
<i>\$HOME/mbox</i>	secondary storage file
<i>/usr/mail/*</i>	post office directory
<i>/usr/lib/mailx/mailx.help*</i>	help message files
<i>/usr/lib/mailx/mailx.rc</i>	optional global start-up file
<i>/tmp/R[emqxsx]*</i>	temporary files

SEE ALSO

ls(1), *mail(1)*, *pg(1)*.

WARNINGS

The **-h**, **-r** and **-U** options can be used only if *mailx* is built with a delivery program other than */bin/mail*.

BUGS

Where *shell-command* is shown as valid, arguments are not always allowed. Experimentation is recommended.

Internal variables imported from the execution environment cannot be unset. The full internet addressing is not fully supported by *mailx*. The new standards need some time to settle down.

Attempts to send a message having a line consisting only of a "." are treated as the end of the message by *mail(1)* (the standard mail delivery program).

NAME

makekey – generate encryption key

SYNOPSIS

/usr/lib/makekey

DESCRIPTION

makekey improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It attempts to read 8 bytes for its *key* (the first eight input bytes), then it attempts to read 2 bytes for its *salt* (the last two input bytes). The output depends on the input in a way intended to be difficult to compute (i.e., to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, *.*, *l*, and upper- and lower-case letters. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4,096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but broken in 4,096 different ways. Using the *input key* as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 *output key* bits in the result.

makekey is intended for programs that perform encryption. Usually, its input and output will be pipes.

SEE ALSO

ed(1), crypt(1), vi(1).
passwd(4) in the *System Administrator's Reference Manual*.

CAVEATS

makekey can produce different results depending upon whether the input is typed at the terminal or redirected from a file.

WARNING

This command is provided with the Security Administration Utilities, which is only available in the United States.



NAME

`mesg` – permit or deny messages

SYNOPSIS

`mesg [-n] [-y]`

DESCRIPTION

mesg with argument `n` forbids messages via *write*(1) by revoking non-user write permission on the user's terminal. *mesg* with argument `y` reinstates permission. All by itself, *mesg* reports the current state without changing it.

FILES

`/dev/tty*`

SEE ALSO

write(1).

DIAGNOSTICS

Exit status is 0 if messages are receivable, 1 if not, 2 on error.



NAME

`mkdir` – make directories

SYNOPSIS

`mkdir` [**-m** mode] [**-p**] dirname ...

DESCRIPTION

`mkdir` creates the named directories in mode 777 (possibly altered by `umask(1)`).

Standard entries in a directory (e.g., the files `.`, for the directory itself, and `..`, for its parent) are made automatically. `mkdir` cannot create these entries by name. Creation of a directory requires write permission in the parent directory.

The owner ID and group ID of the new directories are set to the process's real user ID and group ID, respectively.

Two options apply to `mkdir`:

- m** This option allows users to specify the mode to be used for new directories. Choices for modes can be found in `chmod(1)`.
- p** With this option, `mkdir` creates *dirname* by creating all the non-existing parent directories first.

EXAMPLE

To create the subdirectory structure `ltr/jd/jan`, type:

```
mkdir -p ltr/jd/jan
```

SEE ALSO

`sh(1)`, `rm(1)`, `umask(1)`,
`intro(2)`, `mkdir(2)` in the *Programmer's Reference Manual*.

DIAGNOSTICS

`mkdir` returns exit code 0 if all directories given in the command line were made successfully. Otherwise, it prints a diagnostic and returns non-zero. An error code is stored in `errno`.



NAME

nawk – pattern scanning and processing language

SYNOPSIS

nawk [-F *re*] [*parameter...*] [*'prog'*] [-f *progfile*] [*file...*]

DESCRIPTION

nawk is a new version of *awk* that provides capabilities unavailable in previous versions. This version will become the default version of *awk* in the next major UNIX system release.

The -F *re* option defines the input field separator to be the regular expression *re*.

Parameters, in the form *x=... y=...* may be passed to *nawk*, where *x* and *y* are *nawk* built-in variables (see list below).

nawk scans each input *file* for lines that match any of a set of patterns specified in *prog*. The *prog* string must be enclosed in single quotes (') to protect it from the shell. For each pattern in *prog* there may be an associated action performed when a line of a *file* matches the pattern. The set of pattern-action statements may appear literally as *prog* or in a file specified with the -f *progfile* option.

Input files are read in order; if there are no files, the standard input is read. The file name - means the standard input. Each input line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

An input line is normally made up of fields separated by white space. (This default can be changed by using the FS built-in variable or the -F *re* option.) The fields are denoted \$1, \$2, ...; \$0 refers to the entire line.

A pattern-action statement has the form:

```
pattern { action }
```

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

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

```
expression relop expression
expression matchop regular expression
```

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

```
var in array,
```

or a Boolean combination of these.

The special patterns BEGIN and END may be used to capture control before the first input line has been read and after the last input line has been read respectively.

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

A regular expression may be used to separate fields by using the `-F re` option or by assigning the expression to the built-in variable `FS`. The default is to ignore leading blanks and to separate fields by blanks and/or tab characters. However, if `FS` is assigned a value, leading blanks are no longer ignored.

Other built-in variables include:

<code>ARGC</code>	command line argument count
<code>ARGV</code>	command line argument array
<code>FILENAME</code>	name of the current input file
<code>FNR</code>	ordinal number of the current record in the current file
<code>FS</code>	input field separator regular expression (default blank)
<code>NF</code>	number of fields in the current record
<code>NR</code>	ordinal number of the current record
<code>OFMT</code>	output format for numbers (default <code>%.6g</code>)
<code>OFS</code>	output field separator (default blank)
<code>ORS</code>	output record separator (default new-line)
<code>RS</code>	input record separator (default new-line)

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

```

if ( conditional ) statement [ else statement ]
while ( conditional ) statement
do statement while ( conditional )
for ( expression ; conditional ; expression ) statement
for ( var in array ) statement
delete array[subscript]
break
continue
{ [ statement ] ... }
expression # commonly variable = expression
print [ expression-list ] [ >expression ]
printf format [ , expression-list ] [ >expression ]
next # skip remaining patterns on this input line
exit [expr] # skip the rest of the input; exit status is expr
return [expr]

```

Statements are terminated by semicolons, new-lines, or right braces. An empty expression-list stands for the whole input line. Expressions take on string or numeric values as appropriate, and are built using the operators `+`, `-`, `*`, `/`, `%`, and concatenation (indicated by a blank). The C operators `++`, `--`,

+=, **-=**, ***=**, **/=**, and **%=** are also available in expressions. Variables may be scalars, array elements (denoted $x[i]$), or fields. Variables are initialized to the null string or zero. Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. String constants are quoted (").

The **print** statement prints its arguments on the standard output, or on a file if $>expression$ is present, or on a pipe if $|cmd$ is present. The arguments are separated by the current output field separator and terminated by the output record separator. The **printf** statement formats its expression list according to the format [see *printf(3S)* in the *Programmer's Reference Manual*].

nawk has a variety of built-in functions: arithmetic, string, input/output, and general.

The arithmetic functions are: *atan2*, *cos*, *exp*, *int*, *log*, *rand*, *sin*, *sqrt*, and *srand*. *int* truncates its argument to an integer. *rand* returns a random number between 0 and 1. *srand* (*expr*) sets the seed value for *rand* to *expr* or uses the time of day if *expr* is omitted.

The string functions are:

gsub(*for*, *repl*, *in*) behaves like *sub* (see below), except that it replaces successive occurrences of the regular expression (like the *ed* global substitute command).

index(*s*, *t*) returns the position in string *s* where string *t* first occurs, or 0 if it does not occur at all.

length(*s*) returns the length of its argument taken as a string, or of the whole line if there is no argument.

match(*s*, *re*) returns the position in string *s* where the regular expression *re* occurs, or 0 if it does not occur at all. RSTART is set to the starting position (which is the same as the returned value), and RLENGTH is set to the length of the matched string.

split(*s*, *a*, *fs*) splits the string *s* into array elements *a*[1], *a*[2], *a*[*n*], and returns *n*. The separation is done with the regular expression *fs* or with the field separator FS if *fs* is not given.

sprintf(*fmt*, *expr*, *expr*, ...) formats the expressions according to the *printf(3S)* format given by *fmt* and returns the resulting string.

sub(*for*, *repl*, *in*) substitutes the string *repl* in place of the first instance of the regular expression *for* in string *in* and returns the number of substitutions. If *in* is omitted, *nawk* substitutes in the current record (\$0).

substr(*s*, *m*, *n*) returns the *n*-character substring of *s* that begins at position *m*.

The input/output and general functions are:

close(*filename*) closes the file or pipe named *filename*.

cmd|*getline* pipes the output of *cmd* into *getline*; each successive call to *getline* returns the next line of output from *cmd*.

getline sets **\$0** to the next input record from the current input file.

getline <*file* sets **\$0** to the next record from *file*.

getline *var* sets variable *var* instead.

getline *var* <*file* sets *var* from the next record of *file*.

system(*cmd*) executes *cmd* and returns its exit status.

All forms of *getline* return 1 for successful input, 0 for end of file, and -1 for an error.

nawk also provides user-defined functions. Such functions may be defined (in the pattern position of a pattern-action statement) as

```
function name(args,...) { stmts }
func name(args,...) { stmts }
```

Function arguments are passed by value if scalar and by reference if array name. Argument names are local to the function; all other variable names are global. Function calls may be nested and functions may be recursive. The **return** statement may be used to return a value.

EXAMPLES

Print lines longer than 72 characters:

```
length > 72
```

Print first two fields in opposite order:

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

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

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

Add up first column, print sum and average:

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

Print fields in reverse order:

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

Print all lines between start/stop pairs:

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

Print all lines whose first field is different from previous one:

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

Simulate *echo*(1):

```
BEGIN {
  for (i = 1; i < ARGV; i++)
    printf "%s", ARGV[i]
  printf "\n"
```

```
    exit
}
```

Print file, filling in page numbers starting at 5:

```
    /Page/ { $2 = n++; }
    { print }
```

command line: **nawk -f program n=5 input**

SEE ALSO

grep(1), sed(1).
lex(1), printf(3S) in the *Programmer's Reference Manual*.
Programmer's Guide.

BUGS

Input white space is not preserved on output if fields are involved. There are no explicit conversions between numbers and strings. To force an expression to be treated as a number add 0 to it; to force it to be treated as a string concatenate the null string ("") to it.



NAME

`newform` – change the format of a text file

SYNOPSIS

`newform` [`-s`] [`-itabspec`] [`-otabspec`] [`-bn`] [`-en`] [`-pn`] [`-an`] [`-f`]
 [`-cchar`] [`-ln`] [`files`]

DESCRIPTION

`newform` reads lines from the named *files*, or the standard input if no input file is named, and reproduces the lines on the standard output. Lines are reformatted in accordance with command line options in effect.

Except for `-s`, command line options may appear in any order, may be repeated, and may be intermingled with the optional *files*. Command line options are processed in the order specified. This means that option sequences like “`-e15 -l60`” will yield results different from “`-l60 -e15`”. Options are applied to all *files* on the command line.

`-s` Shears off leading characters on each line up to the first tab and places up to 8 of the sheared characters at the end of the line. If more than 8 characters (not counting the first tab) are sheared, the eighth character is replaced by a * and any characters to the right of it are discarded. The first tab is always discarded.

An error message and program exit will occur if this option is used on a file without a tab on each line. The characters sheared off are saved internally until all other options specified are applied to that line. The characters are then added at the end of the processed line.

For example, to convert a file with leading digits, one or more tabs, and text on each line, to a file beginning with the text, all tabs after the first expanded to spaces, padded with spaces out to column 72 (or truncated to column 72), and the leading digits placed starting at column 73, the command would be:

```
newform -s -i -l -a -e file-name
```

`-itabspec` Input tab specification: expands tabs to spaces, according to the tab specifications given. *Tabspec* recognizes all tab specification forms described in *tabs*(1). In addition, *tabspec* may be `--`, in which `newform` assumes that the tab specification is to be found in the first line read from the standard input (see *fspec*(4)). If no *tabspec* is given, *tabspec* defaults to `-8`. A *tabspec* of `-0` expects no tabs; if any are found, they are treated as `-1`.

`-otabspec` Output tab specification: replaces spaces by tabs, according to the tab specifications given. The tab specifications are the same as for `-itabspec`. If no *tabspec* is given, *tabspec* defaults to `-8`. A *tabspec* of `-0` means that no spaces will be converted to tabs on output.

`-bn` Truncate *n* characters from the beginning of the line when the line length is greater than the effective line length (see `-ln`). Default is to truncate the number of characters necessary to obtain the effective line length. The default value is used when `-b` with no *n* is used. This option can be used to delete the sequence numbers

from a COBOL program as follows:

`newform -l1 -b7 file-name`

- `-en` Same as `-bn` except that characters are truncated from the end of the line.
- `-pn` Prefix *n* characters (see `-ck`) to the beginning of a line when the line length is less than the effective line length. Default is to prefix the number of characters necessary to obtain the effective line length.
- `-an` Same as `-pn` except characters are appended to the end of a line.
- `-f` Write the tab specification format line on the standard output before any other lines are output. The tab specification format line which is printed will correspond to the format specified in the *last* `-o` option. If no `-o` option is specified, the line which is printed will contain the default specification of `-8`.
- `-ck` Change the prefix/append character to *k*. Default character for *k* is a space.
- `-ln` Set the effective line length to *n* characters. If *n* is not entered, `-l` defaults to 72. The default line length without the `-l` option is 80 characters. Note that tabs and backspaces are considered to be one character (use `-i` to expand tabs to spaces).

The `-l1` must be used to set the effective line length shorter than any existing line in the file so that the `-b` option is activated.

DIAGNOSTICS

All diagnostics are fatal.

<i>usage: ...</i>	<i>newform</i> was called with a bad option.
<i>not -s format</i>	There was no tab on one line.
<i>can't open file</i>	Self-explanatory.
<i>internal line too long</i>	A line exceeds 512 characters after being expanded in the internal work buffer.
<i>tabspec in error</i>	A tab specification is incorrectly formatted, or specified tab stops are not ascending.
<i>tabspec indirection illegal</i>	A <i>tabspec</i> read from a file (or standard input) may not contain a <i>tabspec</i> referencing another file (or standard input).

0 - normal execution

1 - for any error

SEE ALSO

`csplit(1)`, `tabs(1)`.

`fspec(4)` in the *System Administrator's Reference Manual*.

BUGS

newform normally only keeps track of physical characters; however, for the `-i` and `-o` options, *newform* will keep track of backspaces in order to line up tabs in the appropriate logical columns.

newform will not prompt the user if a *tabspec* is to be read from the standard input (by use of `-i--` or `-o--`).

If the `-f` option is used, and the last `-o` option specified was `-o--`, and was preceded by either a `-o--` or a `-i--`, the tab specification format line will be incorrect.



NAME

`newgrp` – log in to a new group

SYNOPSIS

`newgrp [-] [group]`

DESCRIPTION

`newgrp` changes a user's group identification. The user remains logged in and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new real and effective group IDs. The user is always given a new shell, replacing the current shell, by `newgrp`, regardless of whether it terminated successfully or due to an error condition (i.e., unknown group).

Exported variables retain their values after invoking `newgrp`; however, all unexported variables are either reset to their default value or set to null. System variables (such as `PS1`, `PS2`, `PATH`, `MAIL`, and `HOME`), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (`PS1`) other than `$` (default) and has not exported `PS1`. After an invocation of `newgrp`, successful or not, their `PS1` will now be set to the default prompt string `$`. Note that the shell command `export` (see `sh(1)`) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, `newgrp` changes the group identification back to the group specified in the user's password file entry. This is a way to exit the effect of an earlier `newgrp` command.

If the first argument to `newgrp` is a `-`, the environment is changed to what would be expected if the user actually logged in again as a member of the new group.

A password is demanded if the group has a password and the user does not, or if the group has a password and the user is not listed in `/etc/group` as being a member of that group.

FILES

<code>/etc/group</code>	system's group file
<code>/etc/passwd</code>	system's password file

SEE ALSO

`login(1)`, `sh(1)` in the *User's Reference Manual*.
`group(4)`, `passwd(4)`, `environ(5)` in the *System Administrator's Reference Manual*.

BUGS

There is no convenient way to enter a password into `/etc/group`. Use of group passwords is not encouraged, because, by their very nature, they encourage poor security practices. Group passwords may disappear in the future.



NAME

`news` – print news items

SYNOPSIS

`news [-a] [-n] [-s] [items]`

DESCRIPTION

`news` is used to keep the user informed of current events. By convention, these events are described by files in the directory `/usr/news`.

When invoked without arguments, `news` prints the contents of all current files in `/usr/news`, most recent first, with each preceded by an appropriate header. `news` stores the “currency” time as the modification date of a file named `.news_time` in the user’s home directory (the identity of this directory is determined by the environment variable `$HOME`); only files more recent than this currency time are considered “current.”

- `-a` option causes `news` to print all items, regardless of currency. In this case, the stored time is not changed.
- `-n` option causes `news` to report the names of the current items without printing their contents, and without changing the stored time.
- `-s` option causes `news` to report how many current items exist, without printing their names or contents, and without changing the stored time. It is useful to include such an invocation of `news` in one’s `.profile` file, or in the system’s `/etc/profile`.

All other arguments are assumed to be specific news items that are to be printed.

If a `delete` is typed during the printing of a news item, printing stops and the next item is started. Another `delete` within one second of the first causes the program to terminate.

FILES

`/etc/profile`
`/usr/news/*`
`$HOME/.news_time`

SEE ALSO

`profile(4)`, `environ(5)` in the *System Administrator’s Reference Manual*.



NAME

`nice` -- run a command at low priority

SYNOPSIS

`nice` [`-increment`] `command` [`arguments`]

DESCRIPTION

nice executes *command* with a lower CPU scheduling priority. If the *increment* argument (in the range 1-19) is given, it is used; if not, an increment of 10 is assumed.

The super-user may run commands with priority higher than normal by using a negative increment, e.g., `--10`.

SEE ALSO

`nohup`(1).

`nice`(2) in the *Programmer's Reference Manual*.

DIAGNOSTICS

nice returns the exit status of the subject command.

BUGS

An *increment* larger than 19 is equivalent to 19.



NAME

nl – line numbering filter

SYNOPSIS

nl [-*h*type] [-*b*type] [-*f*type] [-*v*start#] [-*i*incr] [-*p*] [-*l*num] [-*s*sep] [-*w*width] [-*n*format] [-*d*delim] file

DESCRIPTION

nl reads lines from the named *file* or the standard input if no *file* is named and reproduces the lines on the standard output. Lines are numbered on the left in accordance with the command options in effect.

nl views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page. A logical page consists of a header, a body, and a footer section. Empty sections are valid. Different line numbering options are independently available for header, body, and footer (e.g., no numbering of header and footer lines while numbering blank lines only in the body).

The start of logical page sections are signaled by input lines containing nothing but the following delimiter character(s):

<i>Line contents</i>	<i>Start of</i>
\\: : :	header
\\: :	body
\\:	footer

Unless optioned otherwise, *nl* assumes the text being read is in a single logical page body.

Command options may appear in any order and may be intermingled with an optional file name. Only one file may be named. The options are:

-*b*type Specifies which logical page body lines are to be numbered. Recognized *types* and their meaning are:

-*h*type Same as **-*b*type** except for header. Default *type* for logical page header is **n** (no lines numbered).

a	number all lines
t	number lines with printable text only
n	no line numbering
pstring	number only lines that contain the regular expression specified in <i>string</i> .

Default *type* for logical page body is **t** (text lines numbered).

-*f*type Same as **-*b*type** except for footer. Default for logical page footer is **n** (no lines numbered).

-*v*start# *Start#* is the initial value used to number logical page lines. Default is **1**.

-*i*incr *Incr* is the increment value used to number logical page lines. Default is **1**.

- p** Do not restart numbering at logical page delimiters.
- lnum** *Num* is the number of blank lines to be considered as one. For example, **-12** results in only the second adjacent blank being numbered (if the appropriate **-ha**, **-ba**, and/or **-fa** option is set). Default is 1.
- ssep** *Sep* is the character(s) used in separating the line number and the corresponding text line. Default *sep* is a tab.
- wwidth** *Width* is the number of characters to be used for the line number. Default *width* is 6.
- nformat** *Format* is the line numbering format. Recognized values are: **ln**, left justified, leading zeroes suppressed; **rn**, right justified, leading zeroes suppressed; **rz**, right justified, leading zeroes kept. Default *format* is **rn** (right justified).
- dxx** The delimiter characters specifying the start of a logical page section may be changed from the default characters (\:) to two user-specified characters. If only one character is entered, the second character remains the default character (:). No space should appear between the **-d** and the delimiter characters. To enter a backslash, use two backslashes.

EXAMPLE

The command:

```
nl -v10 -i10 -d!+ file1
```

will number *file1* starting at line number 10 with an increment of ten. The logical page delimiters are **!+**.

SEE ALSO

pr(1).

NAME

`nohup` – run a command immune to hangups and quits

SYNOPSIS

`nohup` *command* [*arguments*]

DESCRIPTION

nohup executes *command* with hangups and quits ignored. If output is not redirected by the user, both standard output and standard error are sent to `nohup.out`. If `nohup.out` is not writable in the current directory, output is redirected to `$HOME/nohup.out`.

EXAMPLE

It is frequently desirable to apply *nohup* to pipelines or lists of commands. This can be done only by placing pipelines and command lists in a single file, called a shell procedure. One can then issue:

```
nohup sh file
```

and the *nohup* applies to everything in *file*. If the shell procedure *file* is to be executed often, then the need to type *sh* can be eliminated by giving *file* execute permission. Add an ampersand and the contents of *file* are run in the background with interrupts also ignored (see *sh(1)*):

```
nohup file &
```

An example of what the contents of *file* could be is:

```
sort ofile > nfile
```

SEE ALSO

`chmod(1)`, `nice(1)`, `sh(1)`,
`signal(2)` in the *Programmer's Reference Manual*.

WARNINGS

In the case of the following command

```
nohup command1; command2
```

nohup applies only to `command1`. The command

```
nohup (command1; command2)
```

is syntactically incorrect.



NAME

od - octal dump

SYNOPSIS

od [-bcdosx] [file] [[+]offset[. [b]]

DESCRIPTION

od dumps *file* in one or more formats as selected by the first argument. If the first argument is missing, -o is default. The meanings of the format options are:

- b Interpret bytes in octal.
- c Interpret bytes in ASCII. Certain non-graphic characters appear as C escapes: null=\0, backspace=\b, form-feed=\f, new-line=\n, return=\r, tab=\t; others appear as 3-digit octal numbers.
- d Interpret words in unsigned decimal.
- o Interpret words in octal.
- s Interpret 16-bit words in signed decimal.
- x Interpret words in hex.

The *file* argument specifies which file is to be dumped. If no file argument is specified, the standard input is used.

The *offset* argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as octal bytes. If . is appended, the offset is interpreted in decimal. If b is appended, the offset is interpreted in blocks of 512 bytes. If the file argument is omitted, the offset argument must be preceded by +.

Dumping continues until end-of-file.



NAME

pack, *pcat*, *unpack* – compress and expand files

SYNOPSIS

pack [-] [-f] *name* ...

pcat *name* ...

unpack *name* ...

DESCRIPTION

pack attempts to store the specified files in a compressed form. Wherever possible (and useful), each input file *name* is replaced by a packed file *name.z* with the same access modes, access and modified dates, and owner as those of *name*. The -f option will force packing of *name*. This is useful for causing an entire directory to be packed even if some of the files will not benefit. If *pack* is successful, *name* will be removed. Packed files can be restored to their original form using *unpack* or *pcat*.

pack uses Huffman (minimum redundancy) codes on a byte-by-byte basis. If the - argument is used, an internal flag is set that causes the number of times each byte is used, its relative frequency, and the code for the byte to be printed on the standard output. Additional occurrences of - in place of *name* will cause the internal flag to be set and reset.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each .z file, it is usually not worthwhile to pack files smaller than three blocks, unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60-75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

pack returns a value that is the number of files that it failed to compress.

No packing will occur if:

- the file appears to be already packed;
- the file name has more than 12 characters;
- the file has links;
- the file is a directory;
- the file cannot be opened;
- no disk storage blocks will be saved by packing;
- a file called *name.z* already exists;
- the .z file cannot be created;
- an I/O error occurred during processing.

The last segment of the file name must contain no more than 12 characters to allow space for the appended .z extension. Directories cannot be compressed.

Pcat does for packed files what *cat*(1) does for ordinary files, except that *pcat* cannot be used as a filter. The specified files are unpacked and written to the standard output. Thus to view a packed file named *name.z* use:

```
pcat name.z  
or just:  
pcat name
```

To make an unpacked copy, say *nnn*, of a packed file named *name.z* (without destroying *name.z*) use the command:

```
pcat name >nnn
```

Pcat returns the number of files it was unable to unpack. Failure may occur if:

- the file name (exclusive of the *.z*) has more than 12 characters;
- the file cannot be opened;
- the file does not appear to be the output of *pack*.

Unpack expands files created by *pack*. For each file *name* specified in the command, a search is made for a file called *name.z* (or just *name*, if *name* ends in *.z*). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the *.z* suffix stripped from its name, and has the same access modes, access and modification dates, and owner as those of the packed file.

Unpack returns a value that is the number of files it was unable to unpack. Failure may occur for the same reasons that it may in *pcat*, as well as for the following:

- a file with the "unpacked" name already exists;
- if the unpacked file cannot be created.

SEE ALSO

cat(1).

NAME

`passwd` – change login password and password attributes

SYNOPSIS

`passwd` [*name*]

`passwd` [`-l` | `-d`] [`-n` *min*] [`-f`] [`-x` *max*] *name*

`passwd` `-s` [`-a`]

`passwd` `-s` [*name*]

DESCRIPTION

The `passwd` command changes the password or lists password attributes associated with the user's login *name*. Additionally, super-users may use `passwd` to install or change passwords and attributes associated with any login *name*.

When used to change a password, `passwd` prompts ordinary users for their old password, if any. It then prompts for the new password twice. When the old password is entered, `passwd` checks to see if it has "aged" sufficiently. If "aging" is insufficient, `passwd` terminates; see `passwd(4)`.

Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical the cycle of prompting for the new password is repeated for at most two more times.

Passwords must be constructed to meet the following requirements:

Each password must have at least six characters. Only the first eight characters are significant.

Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" refers to all upper or lower case letters.

Each password must differ from the user's login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

Super-users (e.g., real and effective uid equal to zero, see `id(1M)` and `su(1M)`) may change any password; hence, `passwd` does not prompt super-users for the old password. Super-users are not forced to comply with password aging and password construction requirements. A super-user can create a null password by entering a carriage return in response to the prompt for a new password. (This differs from `passwd -d` because the "password" prompt will still be displayed.)

Any user may use the `-s` option to show password attributes for his or her own login *name*.

The format of the display will be:

name status mm/dd/yy min max

or, if password aging information is not present,

name status

where

- name* The login ID of the user.
- status* The password status of *name*: "PS" stands for passworded or locked, "LK" stands for locked, and "NP" stands for no password.
- mm/dd/yy* The date password was last changed for *name*. (Note that all password aging dates are determined using Greenwich Mean Time and, therefore, may differ by as much as a day in other time zones.)
- min* The minimum number of days required between password changes for *name*.
- max* The maximum number of days the password is valid for *name*.

Only a super-user can use the following options:

- l Locks password entry for *name*.
- d Deletes password for *name*. The login *name* will not be prompted for password.
- n Set minimum field for *name*. The *min* field contains the minimum number of days between password changes for *name*. If *min* is greater than *max*, the user may not change the password. Always use this option with the -x option, unless *max* is set to -1 (aging turned off). In that case, *min* need not be set.
- x Set maximum field for *name*. The *max* field contains the number of days that the password is valid for *name*. The aging for *name* will be turned off immediately if *max* is set to -1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.
- a Show password attributes for all entries. Use only with -s option; *name* must not be provided.
- f Force the user to change password at the next login by expiring the password for *name*.

FILES

/etc/passwd, /etc/shadow, /etc/opasswd, /etc/oshadow

SEE ALSO

login(1).
 crypt(3C), passwd(4) in the *Programmer's Reference Manual*.
 id(1M), passmgmt(1M), pwconv(1M), su(1M), in the *System Administrator's Reference Manual*.

WARNING

If the optional `/etc/shadow` file feature is used, the `passwd(1)` command will use that, instead of the `/etc/passwd` file, to obtain password information. Since the way password aging information is stored in the two files is slightly different, the output from `passwd` options that use this information may also be different.

DIAGNOSTICS

The `passwd` command exits with one of the following values:

- 0 SUCCESS.
- 1 Permission denied.
- 2 Invalid combination of options.
- 3 Unexpected failure. Password file unchanged.
- 4 Unexpected failure. Password file(s) missing.
- 5 Password file(s) busy. Try again later.
- 6 Invalid argument to option.

3

5

7

NAME

`paste` - merge same lines of several files or subsequent lines of one file

SYNOPSIS

```
paste file1 file2 ...
paste -dlist file1 file2 ...
paste -s [-dlist] file1 file2 ...
```

DESCRIPTION

In the first two forms, *paste* concatenates corresponding lines of the given input files *file1*, *file2*, etc. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). If you will, it is the counterpart of *cat(1)* which concatenates vertically, i.e., one file after the other. In the last form above, *paste* replaces the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the *tab* character, or with characters from an optionally specified *list*. Output is to the standard output, so it can be used as the start of a pipe, or as a filter, if `-` is used in place of a file name.

The meanings of the options are:

- `-d` Without this option, the new-line characters of each but the last file (or last line in case of the `-s` option) are replaced by a *tab* character. This option allows replacing the *tab* character by one or more alternate characters (see below).
- list* One or more characters immediately following `-d` replace the default *tab* as the line concatenation character. The *list* is used circularly, i.e., when exhausted, it is reused. In parallel merging (i.e., no `-s` option), the lines from the last file are always terminated with a new-line character, not from the *list*. The *list* may contain the special escape sequences: `\n` (new-line), `\t` (*tab*), `\\` (backslash), and `\0` (empty string, not a null character). Quoting may be necessary, if characters have special meaning to the shell (e.g., to get one backslash, use `-d"\\\\"`).
- `-s` Merge subsequent lines rather than one from each input file. Use *tab* for concatenation, unless a *list* is specified with `-d` option. Regardless of the *list*, the very last character of the file is forced to be a new-line.
- `-` May be used in place of any file name, to read a line from the standard input. (There is no prompting).

EXAMPLES

```
ls | paste -d" " -           list directory in one column
ls | paste - - - -         list directory in four columns
paste -s -d"\t\n" file     combine pairs of lines into lines
```

SEE ALSO

`cut(1)`, `grep(1)`, `pr(1)`.

PASTE(1)

(Directory and File Management Utilities)

PASTE(1)

DIAGNOSTICS

line too long
too many files

Output lines are restricted to 511 characters.
Except for `-s` option, no more than 12 input files
may be specified.



NAME

pg – file perusal filter for CRTs

SYNOPSIS

pg [*-number*] [*-p string*] [*-cefns*] [*+linenumber*] [*+/pattern/*] [*files...*]

DESCRIPTION

The *pg* command is a filter which allows the examination of *files* one screenful at a time on a CRT. (The file name – and/or NULL arguments indicate that *pg* should read from the standard input.) Each screenful is followed by a prompt. If the user types a carriage return, another page is displayed; other possibilities are enumerated below.

This command is different from previous paginators in that it allows you to back up and review something that has already passed. The method for doing this is explained below.

In order to determine terminal attributes, *pg* scans the *terminfo*(4) data base for the terminal type specified by the environment variable TERM. If TERM is not defined, the terminal type **dumb** is assumed.

The command line options are:

-number

An integer specifying the size (in lines) of the window that *pg* is to use instead of the default. (On a terminal containing 24 lines, the default window size is 23).

-p string

Causes *pg* to use *string* as the prompt. If the prompt string contains a “%d”, the first occurrence of “%d” in the prompt will be replaced by the current page number when the prompt is issued. The default prompt string is “:”.

-c

Home the cursor and clear the screen before displaying each page. This option is ignored if *clear_screen* is not defined for this terminal type in the *terminfo*(4) data base.

-e

Causes *pg* *not* to pause at the end of each file.

-f

Normally, *pg* splits lines longer than the screen width, but some sequences of characters in the text being displayed (e.g., escape sequences for underlining) generate undesirable results. The *-f* option inhibits *pg* from splitting lines.

-n

Normally, commands must be terminated by a *<newline>* character. This option causes an automatic end of command as soon as a command letter is entered.

-s

Causes *pg* to print all messages and prompts in standout mode (usually inverse video).

+linenumber

Start up at *linenumber*.

+/pattern/

Start up at the first line containing the regular expression pattern.

The responses that may be typed when *pg* pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands which cause further perusal normally take a preceding *address*, an optionally signed number indicating the point from which further text should be displayed. This *address* is interpreted in either pages or lines depending on the command. A signed *address* specifies a point relative to the current page or line, and an unsigned *address* specifies an address relative to the beginning of the file. Each command has a default address that is used if none is provided.

The perusal commands and their defaults are as follows:

(+1) <*newline*> or <*blank*>

This causes one page to be displayed. The address is specified in pages.

(+1) **l** With a relative address this causes *pg* to simulate scrolling the screen, forward or backward, the number of lines specified. With an absolute address this command prints a screenful beginning at the specified line.

(+1) **d** or **^D**

Simulates scrolling half a screen forward or backward.

The following perusal commands take no *address*.

. or **^L** Typing a single period causes the current page of text to be redisplayed.

\$ Displays the last windowful in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The regular expressions described in *ed*(1) are available. They must always be terminated by a <*newline*>, even if the *-n* option is specified.

i/*pattern*/

Search forward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately after the current page and continues to the end of the current file, without wrap-around.

i^*pattern*^

i?*pattern*?

Search backwards for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately before the current page and continues to the beginning of the current file, without wrap-around. The ^ notation is useful for Adds 100 terminals which will not properly handle the ?.

After searching, *pg* will normally display the line found at the top of the screen. This can be modified by appending **m** or **b** to the search command to leave the line found in the middle or at the bottom of the window from now on. The suffix **t** can be used to restore the original situation.

The user of *pg* can modify the environment of perusal with the following commands:

in Begin perusing the *i*th next file in the command line. The *i* is an unsigned number, default value is 1.

ip Begin perusing the *i*th previous file in the command line. *i* is an unsigned number, default is 1.

iw Display another window of text. If *i* is present, set the window size to *i*.

s *filename*

Save the input in the named file. Only the current file being perused is saved. The white space between the *s* and *filename* is optional. This command must always be terminated by a *<newline>*, even if the *-n* option is specified.

h Help by displaying an abbreviated summary of available commands.

q or **Q** Quit *pg*.

!*command*

Command is passed to the shell, whose name is taken from the SHELL environment variable. If this is not available, the default shell is used. This command must always be terminated by a *<newline>*, even if the *-n* option is specified.

At any time when output is being sent to the terminal, the user can hit the quit key (normally control-*-*) or the interrupt (break) key. This causes *pg* to stop sending output, and display the prompt. The user may then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

If the standard output is not a terminal, then *pg* acts just like *cat(1)*, except that a header is printed before each file (if there is more than one).

EXAMPLE

A sample usage of *pg* in reading system news would be

```
news | pg -p "(Page %d):"
```

NOTES

While waiting for terminal input, *pg* responds to **BREAK**, **DEL**, and **^** by terminating execution. Between prompts, however, these signals interrupt *pg*'s current task and place the user in prompt mode. These should be used with caution when input is being read from a pipe, since an interrupt is likely to terminate the other commands in the pipeline.

Users of Berkeley's *more* will find that the *z* and *f* commands are available, and that the terminal */*, *^*, or *?* may be omitted from the searching commands.

FILES

/usr/lib/terminfo/?/* terminal information database
/tmp/pg* temporary file when input is from a pipe

SEE ALSO

ed(1), grep(1).
terminfo(4) in the *System Administrator's Reference Manual* .

BUGS

If terminal tabs are not set every eight positions, undesirable results may occur.

When using *pg* as a filter with another command that changes the terminal I/O options terminal settings may not be restored correctly.

NAME

`pr` - print files

SYNOPSIS

```
pr [[-column] [-wwidth] [-a]] [-eck] [-ick] [-drtfp] [+page] [-nck]
[-ooffset] [-llength] [-separator] [-hheader] [file ...]
```

```
pr [[-m] [-wwidth]] [-eck] [-ick] [-drtfp] [+page] [-nck] [-ooffset]
[-lheader] [-separator] [-hheader] file1 file2 ...
```

DESCRIPTION

`pr` is used to format and print the contents of a file. If *file* is `-`, or if no files are specified, `pr` assumes standard input. `pr` prints the named files on standard output.

By default, the listing is separated into pages, each headed by the page number, the date and time that the file was last modified, and the name of the file. Page length is 66 lines which includes 10 lines of header and trailer output. The header is composed of 2 blank lines, 1 line of text (can be altered with `-h`), and 2 blank lines; the trailer is 5 blank lines. For single column output, line width may not be set and is unlimited. For multicolumn output, line width may be set and the default is 72 columns. Diagnostic reports (failed options) are reported at the end of standard output associated with a terminal, rather than interspersed in the output. Pages are separated by series of line feeds rather than form feed characters.

By default, columns are of equal width, separated by at least one space; lines which do not fit are truncated. If the `-s` option is used, lines are not truncated and columns are separated by the *separator* character.

Either `-column` or `-m` should be used to produce multi-column output. `-a` should only be used with `-column` and not `-m`.

Command line options are

- `+page` Begin printing with page numbered *page* (default is 1).
- `-column` Print *column* columns of output (default is 1). Output appears as if `-e` and `-i` are turned on for multi-column output. May not use with `-m`.
- `-a` Print multi-column output across the page one line per column. *columns* must be greater than one. If a line is too long to fit in a column, it is truncated.
- `-m` Merge and print all files simultaneously, one per column. The maximum number of files that may be specified is eight. If a line is too long to fit in a column, it is truncated. May not use with `-column`.
- `-d` Double-space the output. Blank lines that result from double-spacing are dropped when they occur at the top of a page.
- `-eck` Expand input tabs to character positions $k+1$, $2*k+1$, $3*k+1$, etc. If k is 0 or is omitted, default tab settings at every eighth position are assumed. Tab characters in the input are expanded into the

appropriate number of spaces. If *c* (any non-digit character) is given, it is treated as the input tab character (default for *c* is the tab character).

- ick** In output, replace white space wherever possible by inserting tabs to character positions $k+1$, $2*k+1$, $3*k+1$, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any non-digit character) is given, it is treated as the output tab character (default for *c* is the tab character).
- nck** Provide *k*-digit line numbering (default for *k* is 5). The number occupies the first $k+1$ character positions of each column of single column output or each line of **-m** output. If *c* (any non-digit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a tab).
- width** Set the width of a line to *width* character positions (default is 72). This is effective only for multi-column output (**-column** and **-m**). There is no line limit for single column output.
- offset** Offset each line by *offset* character positions (default is 0). The number of character positions per line is the sum of the width and offset.
- length** Set the length of a page to *length* lines (default is 66). **-l0** is reset to **-l66**. When the value of *length* is 10 or less, **-t** appears to be in effect since headers and trailers are suppressed. By default, output contains 5 lines of header and 5 lines of trailer leaving 56 lines for user-supplied text. When **-length** is used and *length* exceeds 10, then $length-10$ lines are left per page for user supplied text. When *length* is 10 or less, header and trailer output is omitted to make room for user supplied text.
- h header** Use *header* as the text line of the header to be printed instead of the file name. **-h** is ignored when **-t** is specified or **-length** is specified and the value of *length* is 10 or less. (**-h** is the only *pr* option requiring space between the option and argument.)
- p** Pause before beginning each page if the output is directed to a terminal (*pr* will ring the bell at the terminal and wait for a carriage return).
- f** Use single form-feed character for new pages (default is to use a sequence of line-feeds). Pause before beginning the first page if the standard output is associated with a terminal.
- r** Print no diagnostic reports on files that will not open.
- t** Print neither the five-line identifying header nor the five-line trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page. Use of **-t** overrides the **-h** option.
- separator** Separate columns by the single character *separator* instead of by the appropriate number of spaces (default for *separator* is a tab).

Prevents truncation of lines on multicolumn output unless `-w` is specified.

EXAMPLES

Print `file1` and `file2` as a double-spaced, three-column listing headed by "file list":

```
pr -3dh "file list" file1 file2
```

Copy `file1` to `file2`, expanding tabs to columns 10, 19, 28, 37, ... :

```
pr -e9 -t <file1 > file2
```

Print `file1` and `file2` simultaneously in a two-column listing with no header or trailer where both columns have line numbers:

```
pr -t -n file1 | pr -t -m -n file2 -
```

FILES

`/dev/tty*` If standard output is directed to one of the special files `/dev/tty*`, then other output directed to this terminal is delayed until standard output is completed. This prevents error messages from being interspersed throughout the output.

SEE ALSO

`cat(1)`, `pg(1)`.



NAME

ps — report process status

SYNOPSIS

ps [options]

DESCRIPTION

ps prints certain information about active processes. Without *options*, information is printed about processes associated with the controlling terminal. The output consists of a short listing containing only the process ID, terminal identifier, cumulative execution time, and the command name. Otherwise, the information that is displayed is controlled by the selection of *options*.

Options accept names or lists as arguments. Arguments can be either separated from one another by commas or enclosed in double quotes and separated from one another by commas or spaces. Values for *proclist* and *grplist* must be numeric.

The *options* are given in descending order according to volume and range of information provided:

- e Print information about every process now running.
- d Print information about all processes except process group leaders.
- a Print information about all processes most frequently requested: all those except process group leaders and processes not associated with a terminal.
- f Generate a full listing. (See below for significance of columns in a full listing.)
- l Generate a long listing. (See below.)
- n *name* Valid only for users with a real user id of *root* or a real group id of *sys*. Takes argument signifying an alternate system *name* in place of */unix*.
- t *termlist* List only process data associated with the terminal given in *termlist*. Terminal identifiers may be specified in one of two forms: the device's file name (e.g., *tty04*) or, if the device's file name starts with *tty*, just the digit identifier (e.g., *04*).
- p *proclist* List only process data whose process ID numbers are given in *proclist*.
- u *uidlist* List only process data whose user ID number or login name is given in *uidlist*. In the listing, the numerical user ID will be printed unless you give the *—f* option, which prints the login name.
- g *grplist* List only process data whose process group leader's ID number(s) appears in *grplist*. (A group leader is a process whose process ID number is identical to its process group ID number. A login shell is a common example of a process group leader.)

Under the *—f* option, *ps* tries to determine the command name and arguments given when the process was created by examining the user block. Failing this, the command name is printed, as it would have appeared without the *—f* option, in square brackets.

The column headings and the meaning of the columns in a *ps* listing are given below; the letters **f** and **l** indicate the option (full or long, respectively) that causes the corresponding heading to appear; **all** means that the heading always appears. Note that these two options determine only what information is provided for a process; they do not determine which processes will be listed.

F	(l)	Flags (hexadecimal and additive) associated with the process
		00 Process has terminated: process table entry now available.
		01 A system process: always in primary memory.
		02 Parent is tracing process.
		04 Tracing parent's signal has stopped process: parent is waiting [<i>ptrace(2)</i>].
		08 Process is currently in primary memory.
		10 Process currently in primary memory: locked until an event completes.
S	(l)	The state of the process:
		O Process is running on a processor.
		S Sleeping: process is waiting for an event to complete.
		R Runnable: process is on run queue.
		I Idle: process is being created.
		Z Zombie state: process terminated and parent not waiting.
		T Traced: process stopped by a signal because parent is tracing it.
		X SXBRK state: process is waiting for more primary memory.
UID	(f,l)	The user ID number of the process owner (the login name is printed under the <i>-f</i> option).
PID	(all)	The process ID of the process (this datum is necessary in order to kill a process).
PPID	(f,l)	The process ID of the parent process.
C	(f,l)	Processor utilization for scheduling.
PRI	(l)	The priority of the process (higher numbers mean lower priority).
NI	(l)	Nice value, used in priority computation.
ADDR	(l)	The memory address of the process.
SZ	(l)	The size (in pages or clicks) of the swappable process's image in main memory.

WCHAN	(l)	The address of an event for which the process is sleeping, or in SXBRK state, (if blank, the process is running).
STIME	(f)	The starting time of the process, given in hours, minutes, and seconds. (A process begun more than twenty-four hours before the <i>ps</i> inquiry is executed is given in months and days.)
TTY	(all)	The controlling terminal for the process (the message, <i>?</i> , is printed when there is no controlling terminal).
TIME	(all)	The cumulative execution time for the process.
COMMAND	(all)	The command name (the full command name and its arguments are printed under the <i>-f</i> option).

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked **<defunct>**.

FILES

/dev	
/dev/sxt/*	
/dev/tty*	
/dev/xt/*	terminal ("tty") names searcher files
/dev/kmem	kernel virtual memory
/dev/swap	the default swap device
/dev/mem	memory
/etc/passwd	UID information supplier
/etc/ps_data	internal data structure
/unix	system namelist

SEE ALSO

kill(1), nice(1).
 getty(1M) in the *System Administrator's Reference Manual*.

WARNING

Things can change while *ps* is running; the snap-shot it gives is only true for a split-second, and it may not be accurate by the time you see it. Some data printed for defunct processes is irrelevant.

If no *termlist*, *proclist*, *uidlist*, or *grplist* is specified, *ps* checks *stdin*, *stdout*, and *stderr* in that order, looking for the controlling terminal and will attempt to report on processes associated with the controlling terminal. In this situation, if *stdin*, *stdout*, and *stderr* are all redirected, *ps* will not find a controlling terminal, so there will be no report.

On a heavily loaded system, *ps* may report an *lseek(2)* error and exit. *ps* may seek to an invalid user area address: having obtained the address of a process' user area, *ps* may not be able to seek to that address before the process exits and the address becomes invalid.

ps -ef may not report the actual start of a tty login session, but rather an earlier time, when a *getty* was last respawned on the tty line.

If the user specifies the *-n* flag, the real and effective UID/GID will be set to the real UID/GID of the user invoking *ps*.



NAME

`pwd` – working directory name

SYNOPSIS

`pwd`

DESCRIPTION

pwd prints the path name of the working (current) directory.

SEE ALSO

`cd(1)`.

DIAGNOSTICS

“Cannot open ..” and “Read error in ..” indicate possible file system trouble and should be referred to a UNIX system administrator.



NAME

relogin – rename login entry to show current layer

SYNOPSIS

`/usr/lib/layer/sys/relogin [-s] [line]`

DESCRIPTION

The *relogin* command changes the terminal *line* field of a user's *utmp*(4) entry to the name of the windowing terminal layer attached to standard input. *write*(1) messages sent to this user are directed to this layer. In addition, the *who*(1) command will show the user associated with this layer. *relogin* may only be invoked under *layers*(1).

relogin is invoked automatically by *layers*(1) to set the *utmp*(4) entry to the terminal line of the first layer created upon startup, and to reset the *utmp*(4) entry to the real line on termination. It may be invoked by a user to designate a different layer to receive *write*(1) messages.

-s Suppress error messages.

line Specifies which *utmp*(4) entry to change. The *utmp*(4) file is searched for an entry with the specified *line* field. That field is changed to the line associated with the standard input. (To learn what lines are associated with a given user, say *jd*oe, type `ps -f -u jd`oe and note the values shown in the TTY field (see *ps*(1))).

FILES

`/etc/utmp` database of users versus terminals

DIAGNOSTICS

Returns **0** upon successful completion, **1** otherwise.

SEE ALSO

layers(1), *mesg*(1), *ps*(1), *who*(1), *write*(1) in the *User's Reference Manual*.
utmp(4) in the *System Administrator's Reference Manual*.

NOTES

If *line* does not belong to the user issuing the *relogin* command or standard input is not associated with a terminal, *relogin* will fail.



NAME

`rm`, `rmdir` – remove files or directories

SYNOPSIS

`rm [-f] [-i] file ...`

`rm -r [-f] [-i] dirname ... [file ...]`

`rmdir [-p] [-s] dirname ...`

DESCRIPTION

`rm` removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. If a file has no write permission and the standard input is a terminal, the full set of permissions (in octal) for the file are printed followed by a question mark. This is a prompt for confirmation. If the answer begins with `y` (for yes), the file is deleted, otherwise the file remains.

Note that if the standard input is not a terminal, the command will operate as if the `-f` option is in effect.

Three options apply to `rm`:

- `-f` This option causes the removal of all files (whether write-protected or not) in a directory without prompting the user. In a write-protected directory, however, files are never removed (whatever their permissions are), but no messages are displayed. If the removal of a write-protected directory is attempted, this option will not suppress an error message.
- `-r` This option causes the recursive removal of any directories and subdirectories in the argument list. The directory will be emptied of files and removed. Note that the user is normally prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting, however, if the `-f` option is used, or if the standard input is not a terminal and the `-i` option is not used.

If the removal of a non-empty, write-protected directory is attempted, the command will always fail (even if the `-f` option is used), resulting in an error message.

- `-i` With this option, confirmation of removal of any write-protected file occurs interactively. It overrides the `-f` option and remains in effect even if the standard input is not a terminal.

Two options apply to `rmdir`:

- `-p` This option allows users to remove the directory `dirname` and its parent directories which become empty. A message is printed on standard output as to whether the whole path is removed or part of the path remains for some reason.
- `-s` This option is used to suppress the message printed on standard error when `-p` is in effect.

DIAGNOSTICS

All messages are generally self-explanatory.

It is forbidden to remove the files "." and ".." in order to avoid the consequences of inadvertently doing something like the following:

rm -r .*

Both *rm* and *rmdir* return exit codes of 0 if all the specified directories are removed successfully. Otherwise, they return a non-zero exit code.

SEE ALSO

unlink(2), *rmdir(2)* in the *Programmer's Reference Manual*.

NAME

sag – system activity graph

SYNOPSIS

sag [options]

DESCRIPTION

sag graphically displays the system activity data stored in a binary data file by a previous *sar*(1) run. Any of the *sar* data items may be plotted singly, or in combination; as cross plots, or versus time. Simple arithmetic combinations of data may be specified. *sag* invokes *sar* and finds the desired data by string-matching the data column header (run *sar* to see what is available). These *options* are passed through to *sar*:

- s *time* Select data later than *time* in the form hh[:mm]. Default is 08:00.
- e *time* Select data up to *time*. Default is 18:00.
- i *sec* Select data at intervals as close as possible to *sec* seconds.
- f *file* Use *file* as the data source for *sar*. Default is the current daily data file `/usr/adm/sa/sadd`.

Other *options*:

- T *term* Produce output suitable for terminal *term*. See *tplot*(1G) for known terminals. Default for *term* is \$TERM.
- x *spec* x axis specification with *spec* in the form:
"name[op name]...[lo hi]"
- y *spec* y axis specification with *spec* in the same form as above.

Name is either a string that will match a column header in the *sar* report, with an optional device name in square brackets, e.g., `r+w/s[dsk-1]`, or an integer value. *Op* is + – * or / surrounded by blanks. Up to five names may be specified. Parentheses are not recognized. Contrary to custom, + and – have precedence over * and /. Evaluation is left to right. Thus `A / A + B * 100` is evaluated $(A/(A+B))*100$, and `A + B / C + D` is $(A+B)/(C+D)$. *Lo* and *hi* are optional numeric scale limits. If unspecified, they are deduced from the data.

A single *spec* is permitted for the x axis. If unspecified, *time* is used. Up to 5 *spec*'s separated by ; may be given for –y. Enclose the –x and –y arguments in " " if they include whitespace. The –y default is:

```
–y "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"
```

EXAMPLES

To see today's CPU utilization:

```
sag
```

To see activity over 15 minutes of all disk drives:

```
TS=date +%H:%M
sar -o tempfile 60 15
TE=date +%H:%M
sag -f tempfile -s $TS -e $TE -y "r+w/s[dsk]"
```

FILES

/usr/adm/sa/sadd daily data file for day *dd*.

SEE ALSO

sar(1), *tplot(1G)*.

NAME

sar – system activity reporter

SYNOPSIS

sar [**-ubdycwaqvmprDSAC**] [**-o file**] **t** [**n**]

sar [**-ubdycwaqvmprDSAC**] [**-s time**] [**-e time**] [**-i sec**] [**-f file**]

DESCRIPTION

sar, in the first instance, samples cumulative activity counters in the operating system at *n* intervals of *t* seconds, where *t* should be 5 or greater. (If the sampling interval is less than 5, the activity of *sar* itself may affect the sample.) If the **-o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second instance, with no sampling interval specified, *sar* extracts data from a previously recorded *file*, either the one specified by the **-f** option or, by default, the standard system activity daily data file `/usr/adm/sa/sadd` for the current day *dd*. The starting and ending times of the report can be bounded via the **-s** and **-e time** arguments of the form *hh[:mm[:ss]]*. The **-i** option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by option:

-u Report CPU utilization (the default):

%usr, *%sys*, *%wio*, *%idle* – portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle. When used with **-D**, *%sys* is split into percent of time servicing requests from remote machines (*%sys remote*) and all other system time (*%sys local*). If you are using a 3B2 Computer with a co-processor the CPU utilization (default) report will contain the following fields:

%usr, *%sys*, *%idle*, *scall/s* – where *scalls/s* is the number of system calls, of all types, encountered on the co-processor per second.

-b Report buffer activity:

bread/s, *bwrit/s* – transfers per second of data between system buffers and disk or other block devices;

lread/s, *lwrit/s* – accesses of system buffers;

%rcache, *%wcache* – cache hit ratios, i. e., $(1 - \text{bread}/\text{lread})$ as a percentage;

pread/s, *pwrit/s* – transfers via raw (physical) device mechanism. When used with **-D**, buffer caching is reported for locally-mounted remote resources.

-d Report activity for each block device, e. g., disk or tape drive, with the exception of XDC disks and tape drives. When data is displayed, the device specification *dsk-* is generally used to represent a disk drive. The device specification used to represent a tape drive is machine dependent. The activity data reported is:

%busy, *avque* – portion of time device was busy servicing a transfer request, average number of requests outstanding during that time;

r+w/s, *blks/s* – number of data transfers from or to device, number of bytes transferred in 512-byte units;

avwait, avserv – average time in ms. that transfer requests wait idly on queue, and average time to be serviced (which for disks includes seek, rotational latency and data transfer times).

- y Report TTY device activity:
rawch/s, canch/s, outh/s – input character rate, input character rate processed by canon, output character rate;
rcvin/s, xmtin/s, mdmin/s – receive, transmit and modem interrupt rates.
- c Report system calls:
scall/s – system calls of all types;
sread/s, swrit/s, fork/s, exec/s – specific system calls;
rchar/s, wchar/s – characters transferred by read and write system calls. When used with **-D**, the system calls are split into incoming, outgoing, and strictly local calls.
- w Report system swapping and switching activity:
swpin/s, swpot/s, bswin/s, bswot/s – number of transfers and number of 512-byte units transferred for swapins and swapouts (including initial loading of some programs);
pswch/s – process switches.
- a Report use of file access system routines:
iget/s, namei/s, dirblk/s.
- q Report average queue length while occupied, and % of time occupied:
runq-sz, %runocc – run queue of processes in memory and runnable;
swpq-sz, %swpocc – swap queue of processes swapped out but ready to run.
- v Report status of process, i-node, file tables:
text-sz, proc-sz, inod-sz, file-sz, lock-sz – entries/size for each table, evaluated once at sampling point;
ov – overflows that occur between sampling points for each table.
- m Report message and semaphore activities:
msg/s, sema/s – primitives per second.
- p Report paging activities:
vflt/s – address translation page faults (valid page not in memory);
pflt/s – page faults from protection errors (illegal access to page) or "copy-on-writes";
pgfil/s – vflt/s satisfied by page-in from file system;
rclm/s – valid pages reclaimed for free list.
- r Report unused memory pages and disk blocks:
freemem – average pages available to user processes;
freeswap – disk blocks available for process swapping.
- D Report Remote File Sharing activity:
When used in combination with **-u**, **-b** or **-c**, it causes **sar** to produce the remote file sharing version of the corresponding report. **-Du** is assumed when only **-D** is specified.

- S Report server and request queue status:
Average number of Remote File Sharing servers on the system (*serv/lo-hi*), % of time receive descriptors are on the request queue (*request %busy*), average number of receive descriptors waiting for service when queue is occupied (*request avg lgth*), % of time there are idle servers (*server %avail*), average number of idle servers when idle ones exist (*server avg avail*).
- A Report all data. Equivalent to `-udqbwcaymprSDC`.
- C Report Remote File Sharing buffer caching overhead:
snd-inv/s - number of invalidation messages per second sent by your machine as a server.
snd-msg/s - total outgoing RFS messages sent per second.
rcv-inv/s - number of invalidation messages received from the remote server.
rcv-msg/s - total number of incoming RFS messages received per second.
dis-bread/s - number of buffer reads that would be eligible for caching if caching were not turned off. (Indicates the penalty of running uncached.)
blk-inv/s - number of buffers removed from the client cache.

EXAMPLES

To see today's CPU activity so far:

```
sar
```

To watch CPU activity evolve for 10 minutes and save data:

```
sar -o temp 60 10
```

To later review disk and tape activity from that period:

```
sar -d -f temp
```

FILES

`/usr/adm/sa/sadd` daily data file, where *dd* are digits representing the day of the month.

SEE ALSO

`sag(1G)`, `sar(1M)`.



NAME

`sdiff` – side-by-side difference program

SYNOPSIS

`sdiff` [options ...] *file1* *file2*

DESCRIPTION

`sdiff` uses the output of `diff(1)` to produce a side-by-side listing of two files indicating those lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical, a `<` in the gutter if the line only exists in *file1*, a `>` in the gutter if the line only exists in *file2*, and a `|` for lines that are different.

For example:

```

x      |      y
a      |      a
b      <
c      <
d      |      d
          >      c

```

The following options exist:

- `-w n` Use the next argument, *n*, as the width of the output line. The default line length is 130 characters.
- `-l` Only print the left side of any lines that are identical.
- `-s` Do not print identical lines.
- `-o output` Use the next argument, *output*, as the name of a third file that is created as a user-controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by `diff(1)`, are printed; where a set of differences share a common gutter character. After printing each set of differences, `sdiff` prompts the user with a `%` and waits for one of the following user-typed commands:

```

l      append the left column to the output file
r      append the right column to the output file
s      turn on silent mode; do not print identical lines
v      turn off silent mode
e l    call the editor with the left column
e r    call the editor with the right column
e b    call the editor with the concatenation of left and
        right
e      call the editor with a zero length file
q      exit from the program

```

On exit from the editor, the resulting file is concatenated on the end of the *output* file.

SDIFF(1)

(Directory and File Management Utilities)

SDIFF(1)

SEE ALSO

diff(1), ed(1).

NAME

sed – stream editor

SYNOPSIS

sed [-n] [-e script] [-f sfile] [files]

DESCRIPTION

sed copies the named *files* (standard input default) to the standard output, edited according to a script of commands. The `-f` option causes the script to be taken from file *sfile*; these options accumulate. If there is just one `-e` option and no `-f` options, the flag `-e` may be omitted. The `-n` option suppresses the default output. A script consists of editing commands, one per line, of the following form:

[address [, address]] function [arguments]

In normal operation, *sed* cyclically copies a line of input into a *pattern space* (unless there is something left after a **D** command), applies in sequence all commands whose *addresses* select that pattern space, and at the end of the script copies the pattern space to the standard output (except under `-n`) and deletes the pattern space.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval.

An *address* is either a decimal number that counts input lines cumulatively across files, a **\$** that addresses the last line of input, or a context address, i.e., a */regular expression/* in the style of *ed(1)* modified thus:

In a context address, the construction *?regular expression?*, where *?* is any character, is identical to */regular expression/*. Note that in the context address *\abc\xdex*, the second *x* stands for itself, so that the regular expression is *abcxdex*.

The escape sequence *\n* matches a new-line *embedded* in the pattern space.

A period *.* matches any character except the *terminal* new-line of the pattern space.

A command line with no addresses selects every pattern space.

A command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

Editing commands can be applied only to non-selected pattern spaces by use of the negation function **!** (below).

In the following list of functions the maximum number of permissible addresses for each function is indicated in parentheses.

The *text* argument consists of one or more lines, all but the last of which end with ** to hide the new-line. Backslashes in text are treated like backslashes in

the replacement string of an *s* command, and may be used to protect initial blanks and tabs against the stripping that is done on every script line. The *rfile* or *wfile* argument must terminate the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

- (1)a\ *text* Append. Place *text* on the output before reading the next input line.
- (2)b *label* Branch to the : command bearing the *label*. If *label* is empty, branch to the end of the script.
- (2)c\ *text* Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle.
- (2)d Delete the pattern space. Start the next cycle.
- (2)D Delete the initial segment of the pattern space through the first new-line. Start the next cycle.
- (2)g Replace the contents of the pattern space by the contents of the hold space.
- (2)G Append the contents of the hold space to the pattern space.
- (2)h Replace the contents of the hold space by the contents of the pattern space.
- (2)H Append the contents of the pattern space to the hold space.
- (1)i\ *text* Insert. Place *text* on the standard output.
- (2)l List the pattern space on the standard output in an unambiguous form. Non-printable characters are displayed in octal notation and long lines are folded.
- (2)n Copy the pattern space to the standard output. Replace the pattern space with the next line of input.
- (2)N Append the next line of input to the pattern space with an embedded new-line. (The current line number changes.)
- (2)p Print. Copy the pattern space to the standard output.
- (2)P Copy the initial segment of the pattern space through the first new-line to the standard output.
- (1)q Quit. Branch to the end of the script. Do not start a new cycle.
- (2)r *rfile* Read the contents of *rfile*. Place them on the output before reading the next input line.
- (2)s/*regular expression*/*replacement*/*flags* Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character may be used instead of /. For a fuller description see *ed*(1). *Flags* is zero or more of:
 - n n= 1 - 512. Substitute for just the n th occurrence of the *regular expression*.
 - g Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.
 - p Print the pattern space if a replacement was made.

- w** *wfile* Write. Append the pattern space to *wfile* if a replacement was made.
- (2)**t** *label* Test. Branch to the **:** command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a **t**. If *label* is empty, branch to the end of the script.
- (2)**w** *wfile* Write. Append the pattern space to *wfile*.
- (2)**x** Exchange the contents of the pattern and hold spaces.
- (2)**y**/*string1*/*string2*/
Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.
- (2)**!** *function* Don't. Apply the *function* (or group, if *function* is **{**) only to lines not selected by the address(es).
- (0)**:** *label* This command does nothing; it bears a *label* for **b** and **t** commands to branch to.
- (1)**=** Place the current line number on the standard output as a line.
- (2)**{** Execute the following commands through a matching **}** only when the pattern space is selected.
- (0) An empty command is ignored.
- (0)**#** If a **#** appears as the first character on the first line of a script file, then that entire line is treated as a comment, with one exception. If the character after the **#** is an **'n'**, then the default output will be suppressed. The rest of the line after **#n** is also ignored. A script file must contain at least one non-comment line.

SEE ALSO

awk(1), ed(1), grep(1).



NAME

setup – initialize system for first user

SYNOPSIS

setup

DESCRIPTION

The *setup* command, which is also accessible as a login by the same name, allows the first user to be established as the "owner" of the machine.

The user is permitted to add the first logins to the system, usually starting with his or her own.

The user can then protect the system from unauthorized modification of the machine configuration and software by giving passwords to the administrative and maintenance functions. Normally, the first user of the machine enters this command through the setup login, which initially has no password, and then gives passwords to the various functions in the system. Any that the user leaves without password protection can be exercised by anyone.

The user can then give passwords to system logins such as "root", "bin", etc. (*provided they do not already have passwords*). Once given a password, each login can only be changed by that login or "root".

The user can then set the date, time and time zone of the machine.

The user can then set the node name of the machine.

SEE ALSO

passwd(1).

DIAGNOSTICS

The *passwd*(1) command complains if the password provided does not meet its standards.

WARNING

If the setup login is not under password control, anyone can put passwords on the other functions.



NAME

sh, rsh – shell, the standard/restricted command programming language

SYNOPSIS

```
sh [ -acefhiknrstuvx ] [ args ]
rsh [ -acefhiknrstuvx ] [ args ]
```

DESCRIPTION

sh is a command programming language that executes commands read from a terminal or a file. *rsh* is a restricted version of the standard command interpreter *sh*; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. See “Invocation” below for the meaning of arguments to the shell.

Definitions

A *blank* is a tab or a space. A *name* is a sequence of letters, digits, or underscores beginning with a letter or underscore. A *parameter* is a name, a digit, or any of the characters *, @, #, ?, -, \$, and !.

Commands

A *simple-command* is a sequence of non-blank *words* separated by *blanks*. The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see *exec(2)*). The *value* of a *simple-command* is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally (see *signal(2)* for a list of status values).

A *pipeline* is a sequence of one or more *commands* separated by |. The standard output of each command but the last is connected by a *pipe(2)* to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command.

A *list* is a sequence of one or more pipelines separated by ;, &, &&, or | |, and optionally terminated by ; or &. Of these four symbols, ; and & have equal precedence, which is lower than that of && and | |. The symbols && and | | also have equal precedence. A semicolon (;) causes sequential execution of the preceding pipeline; an ampersand (&) causes asynchronous execution of the preceding pipeline (i.e., the shell does *not* wait for that pipeline to finish). The symbol && (| |) causes the *list* following it to be executed only if the preceding pipeline returns a zero (non-zero) exit status. An arbitrary number of new-lines may appear in a *list*, instead of semicolons, to delimit commands.

A *command* is either a *simple-command* or one of the following. Unless otherwise stated, the value returned by a command is that of the last *simple-command* executed in the command.

for *name* [**in** *word* ...] **do** *list* **done**

Each time a **for** command is executed, *name* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, then the **for** command executes the **do** *list* once for each positional parameter that is set (see *Parameter Substitution* below). Execution ends when there are no more words in the list.

case *word* **in** [*pattern* [| *pattern*] ...] *list* ;;] ... **esac**

A **case** command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the same as that used for file-name generation (see "File Name Generation") except that a slash, a leading dot, or a dot immediately following a slash need not be matched explicitly.

if *list* **then** *list* [**elif** *list* **then** *list*] ... [**else** *list*] **fi**

The *list* following **if** is executed and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else** *list* is executed. If no **else** *list* or **then** *list* is executed, then the **if** command returns a zero exit status.

while *list* **do** *list* **done**

A **while** command repeatedly executes the **while** *list* and, if the exit status of the last command in the *list* is zero, executes the **do** *list*; otherwise the loop terminates. If no commands in the **do** *list* are executed, then the **while** command returns a zero exit status; **until** may be used in place of **while** to negate the loop termination test.

(*list*)

Execute *list* in a sub-shell.

{*list*;}

list is executed in the current (that is, parent) shell.

name () {*list*;}

Define a function which is referenced by *name*. The body of the function is the *list* of commands between { and }. Execution of functions is described below (see *Execution*).

The following words are only recognized as the first word of a command and when not quoted:

if then else elif fi case esac for while until do done { }

Comments

A word beginning with # causes that word and all the following characters up to a new-line to be ignored.

Command Substitution

The shell reads commands from the string between two grave accents (``) and the standard output from these commands may be used as all or part of a word. Trailing new-lines from the standard output are removed.

No interpretation is done on the string before the string is read, except to remove backslashes (\) used to escape other characters. Backslashes may be used to escape a grave accent (`) or another backslash (\) and are removed before the command string is read. Escaping grave accents allows nested command substitution. If the command substitution lies within a pair of double quotes (" ... ` ... ` ... "), a backslash used to escape a double quote (\") will be removed; otherwise, it will be left intact.

If a backslash is used to escape a new-line character (\new-line), both the backslash and the new-line are removed (see the later section on "Quoting"). In addition, backslashes used to escape dollar signs (\\$) are removed. Since

no interpretation is done on the command string before it is read, inserting a backslash to escape a dollar sign has no effect. Backslashes that precede characters other than \, ', ", new-line, and \$ are left intact when the command string is read.

Parameter Substitution

The character \$ is used to introduce substitutable *parameters*. There are two types of parameters, positional and keyword. If *parameter* is a digit, it is a positional parameter. Positional parameters may be assigned values by **set**. Keyword parameters (also known as variables) may be assigned values by writing:

```
name=value [ name=value ] ...
```

Pattern-matching is not performed on *value*. There cannot be a function and a variable with the same *name*.

\${parameter}

The value, if any, of the parameter is substituted. The braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. If *parameter* is * or @, all the positional parameters, starting with \$1, are substituted (separated by spaces). Parameter \$0 is set from argument zero when the shell is invoked.

\${parameter:-word}

If *parameter* is set and is non-null, substitute its value; otherwise substitute *word*.

\${parameter:=word}

If *parameter* is not set or is null set it to *word*; the value of the parameter is substituted. Positional parameters may not be assigned to in this way.

\${parameter:?word}

If *parameter* is set and is non-null, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, the message "parameter null or not set" is printed.

\${parameter:+word}

If *parameter* is set and is non-null, substitute *word*; otherwise substitute nothing.

In the above, *word* is not evaluated unless it is to be used as the substituted string, so that, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-`pwd`}
```

If the colon (:) is omitted from the above expressions, the shell only checks whether *parameter* is set or not.

The following parameters are automatically set by the shell:

- # The number of positional parameters in decimal.
- Flags supplied to the shell on invocation or by the **set** command.

- ? The decimal value returned by the last synchronously executed command.
- \$ The process number of this shell.
- ! The process number of the last background command invoked.

The following parameters are used by the shell:

- HOME The default argument (home directory) for the *cd* command.
- PATH The search path for commands (see *Execution* below). The user may not change PATH if executing under *rsh*.

CDPATH

The search path for the *cd* command.

- MAIL If this parameter is set to the name of a mail file *and* the MAILPATH parameter is not set, the shell informs the user of the arrival of mail in the specified file.

MAILCHECK

This parameter specifies how often (in seconds) the shell will check for the arrival of mail in the files specified by the MAILPATH or MAIL parameters. The default value is 600 seconds (10 minutes). If set to 0, the shell will check before each prompt.

MAILPATH

A colon (:) separated list of file names. If this parameter is set, the shell informs the user of the arrival of mail in any of the specified files. Each file name can be followed by % and a message that will be printed when the modification time changes. The default message is *you have mail*.

- PS1 Primary prompt string, by default "\$ "
- PS2 Secondary prompt string, by default "> "
- IFS Internal field separators, normally **space**, **tab**, and **new-line**.

SHACCT

If this parameter is set to the name of a file writable by the user, the shell will write an accounting record in the file for each shell procedure executed.

- SHELL When the shell is invoked, it scans the environment (see "Environment" below) for this name. If it is found and 'rsh' is the file name part of its value, the shell becomes a restricted shell.

The shell gives default values to PATH, PS1, PS2, MAILCHECK and IFS. HOME and MAIL are set by *login*(1).

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in IFS) and split into distinct arguments where such characters are found. Explicit null arguments (" or '') are retained. Implicit null arguments (those resulting from *parameters* that have no values) are removed.

Input/Output

A command's input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a *simple-command* or may precede or follow a *command* and are *not* passed on as arguments to the invoked command. Note that parameter and command substitution occurs before *word* or *digit* is used.

- <word** Use file *word* as standard input (file descriptor 0).
- >word** Use file *word* as standard output (file descriptor 1). If the file does not exist it is created; otherwise, it is truncated to zero length.
- >>word** Use file *word* as standard output. If the file exists output is appended to it (by first seeking to the end-of-file); otherwise, the file is created.
- <<[-]word** After parameter and command substitution is done on *word*, the shell input is read up to the first line that literally matches the resulting *word*, or to an end-of-file. If, however, - is appended to <<:
 - 1) leading tabs are stripped from *word* before the shell input is read (but after parameter and command substitution is done on *word*),
 - 2) leading tabs are stripped from the shell input as it is read and before each line is compared with *word*, and
 - 3) shell input is read up to the first line that literally matches the resulting *word*, or to an end-of-file.

If any character of *word* is quoted (see "Quoting," later), no additional processing is done to the shell input. If no characters of *word* are quoted:

 - 1) parameter and command substitution occurs,
 - 2) (escaped) \new-line is ignored, and
 - 3) \ must be used to quote the characters \, \$, and `.

The resulting document becomes the standard input.
- <&digit** Use the file associated with file descriptor *digit* as standard input. Similarly for the standard output using **>&digit**.
- <&-** The standard input is closed. Similarly for the standard output using **>&-**.

If any of the above is preceded by a digit, the file descriptor which will be associated with the file is that specified by the digit (instead of the default 0 or 1). For example:

```
... 2>&1
```

associates file descriptor 2 with the file currently associated with file descriptor 1.

The order in which redirections are specified is significant. The shell evaluates redirections left-to-right. For example:

```
... 1>xxx 2>&1
```

first associates file descriptor 1 with file *xxx*. It associates file descriptor 2 with the file associated with file descriptor 1 (i.e., *xxx*). If the order of redirections were reversed, file descriptor 2 would be associated with the terminal (assuming file descriptor 1 had been) and file descriptor 1 would be associated with file *xxx*.

Using the terminology introduced on the first page, under “Commands,” if a *command* is composed of several *simple commands*, redirection will be evaluated for the entire *command* before it is evaluated for each *simple command*. That is, the shell evaluates redirection for the entire *list*, then each *pipeline* within the *list*, then each *command* within each *pipeline*, then each *list* within each *command*.

If a command is followed by **&** the default standard input for the command is the empty file */dev/null*. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Redirection of output is not allowed in the restricted shell.

File Name Generation

Before a command is executed, each command *word* is scanned for the characters *****, **?**, and **[**. If one of these characters appears the word is regarded as a *pattern*. The word is replaced with alphabetically sorted file names that match the pattern. If no file name is found that matches the pattern, the word is left unchanged. The character **.** at the start of a file name or immediately following a **/**, as well as the character **/** itself, must be matched explicitly.

- *** Matches any string, including the null string.
- ?** Matches any single character.
- [...]** Matches any one of the enclosed characters. A pair of characters separated by **-** matches any character lexically between the pair, inclusive. If the first character following the opening **[** is a **"** any character not enclosed is matched.

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

```
; & ( ) | ^ < > new-line space tab
```

A character may be *quoted* (i.e., made to stand for itself) by preceding it with a backslash (****) or inserting it between a pair of quote marks (**"** or **"**). During processing, the shell may quote certain characters to prevent them from taking on a special meaning. Backslashes used to quote a single character are removed from the word before the command is executed. The pair **\new-line** is removed from a word before command and parameter substitution.

All characters enclosed between a pair of single quote marks (**'**), except a single quote, are quoted by the shell. Backslash has no special meaning inside

a pair of single quotes. A single quote may be quoted inside a pair of double quote marks (for example, `"'"`).

Inside a pair of double quote marks (`"`), parameter and command substitution occurs and the shell quotes the results to avoid blank interpretation and file name generation. If `$*` is within a pair of double quotes, the positional parameters are substituted and quoted, separated by quoted spaces (`"$1 $2 ..."`); however, if `$@` is within a pair of double quotes, the positional parameters are substituted and quoted, separated by unquoted spaces (`"$1" "$2" ...`). `\` quotes the characters `\`, `'`, `"`, and `$`. The pair `\new-line` is removed before parameter and command substitution. If a backslash precedes characters other than `\`, `'`, `"`, `$`, and new-line, then the backslash itself is quoted by the shell.

Prompting

When used interactively, the shell prompts with the value of `PS1` before reading a command. If at any time a new-line is typed and further input is needed to complete a command, the secondary prompt (i.e., the value of `PS2`) is issued.

Environment

The *environment* (see `environ(5)`) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. If the user modifies the value of any of these parameters or creates new parameters, none of these affects the environment unless the `export` command is used to bind the shell's parameter to the environment (see also `set -a`). A parameter may be removed from the environment with the `unset` command. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, minus any pairs removed by `unset`, plus any modifications or additions, all of which must be noted in `export` commands.

The environment for any *simple-command* may be augmented by prefixing it with one or more assignments to parameters. Thus:

```
TERM=450 cmd                                and
(exports TERM; TERM=450; cmd)
```

are equivalent (as far as the execution of `cmd` is concerned).

If the `-k` flag is set, *all* keyword arguments are placed in the environment, even if they occur after the command name. The following first prints `a=b c` and `c`:

```
echo a=b c
set -k
echo a=b c
```

Signals

The `INTERRUPT` and `QUIT` signals for an invoked command are ignored if the command is followed by `&`; otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the `trap` command below).

Execution

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the *Special Commands* listed below, it is executed in the shell process. If the command name does not match a *Special Command*, but matches the name of a defined function, the function is executed in the shell process (note how this differs from the execution of shell procedures). The positional parameters **\$1**, **\$2**, ... are set to the arguments of the function. If the command name matches neither a *Special Command* nor the name of a defined function, a new process is created and an attempt is made to execute the command via *exec(2)*.

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is **:/bin:/usr/bin** (specifying the current directory, **/bin**, and **/usr/bin**, in that order). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign, between two colon delimiters anywhere in the path list, or at the end of the path list. If the command name contains a **/** the search path is not used; such commands will not be executed by the restricted shell. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not an **a.out** file, it is assumed to be a file containing shell commands. A sub-shell is spawned to read it. A parenthesized command is also executed in a sub-shell.

The location in the search path where a command was found is remembered by the shell (to help avoid unnecessary *execs* later). If the command was found in a relative directory, its location must be re-determined whenever the current directory changes. The shell forgets all remembered locations whenever the **PATH** variable is changed or the **hash -r** command is executed (see below).

Special Commands

Input/output redirection is now permitted for these commands. File descriptor 1 is the default output location.

: No effect; the command does nothing. A zero exit code is returned.

. file Read and execute commands from *file* and return. The search path specified by **PATH** is used to find the directory containing *file*.

break [n]

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

continue [n]

Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified resume at the *n*-th enclosing loop.

cd [arg]

Change the current directory to *arg*. The shell parameter **HOME** is the default *arg*. The shell parameter **CDPATH** defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (:). The default path is **<null>** (specifying the current directory). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or

between the colon delimiters anywhere else in the path list. If *arg* begins with a / the search path is not used. Otherwise, each directory in the path is searched for *arg*. The *cd* command may not be executed by *rsh*.

echo [*arg ...*]

Echo arguments. See *echo(1)* for usage and description.

eval [*arg ...*]

The arguments are read as input to the shell and the resulting command(s) executed.

exec [*arg ...*]

The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.

exit [*n*]

Causes a shell to exit with the exit status specified by *n*. If *n* is omitted the exit status is that of the last command executed (an end-of-file will also cause the shell to exit.)

export [*name ...*]

The given *names* are marked for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, variable names that have been marked for export during the current shell's execution are listed. (Variable names exported from a parent shell are listed only if they have been exported again during the current shell's execution.) Function names are *not* exported.

getopts

Use in shell scripts to support command syntax standards (see *intro(1)*); it parses positional parameters and checks for legal options. See *getopts(1)* for usage and description.

hash [*-r*] [*name ...*]

For each *name*, the location in the search path of the command specified by *name* is determined and remembered by the shell. The *-r* option causes the shell to forget all remembered locations. If no arguments are given, information about remembered commands is presented. *Hits* is the number of times a command has been invoked by the shell process. *Cost* is a measure of the work required to locate a command in the search path. If a command is found in a "relative" directory in the search path, after changing to that directory, the stored location of that command is recalculated. Commands for which this will be done are indicated by an asterisk (*) adjacent to the *hits* information. *Cost* will be incremented when the recalculation is done.

newgrp [*arg ...*]

Equivalent to `exec newgrp arg ...`. See *newgrp(1)* for usage and description.

pwd

Print the current working directory. See *pwd(1)* for usage and description.

read [*name* ...]

One line is read from the standard input and, using the internal field separator, IFS (normally space or tab), to delimit word boundaries, the first word is assigned to the first *name*, the second word to the second *name*, etc., with leftover words assigned to the last *name*. Lines can be continued using **\new-line**. Characters other than **\new-line** can be quoted by preceding them with a backslash. These backslashes are removed before words are assigned to *names*, and no interpretation is done on the character that follows the backslash. The return code is 0 unless an end-of-file is encountered.

readonly [*name* ...]

The given *names* are marked *readonly* and the values of these *names* may not be changed by subsequent assignment. If no arguments are given, a list of all *readonly* names is printed.

return [*n*]

Causes a function to exit with the return value specified by *n*. If *n* is omitted, the return status is that of the last command executed.

set [**--aefhkntuvx** [*arg* ...]]

- a** Mark variables which are modified or created for export.
- e** Exit immediately if a command exits with a non-zero exit status.
- f** Disable file name generation
- h** Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).
- k** All keyword arguments are placed in the environment for a command, not just those that precede the command name.
- n** Read commands but do not execute them.
- t** Exit after reading and executing one command.
- u** Treat unset variables as an error when substituting.
- v** Print shell input lines as they are read.
- x** Print commands and their arguments as they are executed.
- Do not change any of the flags; useful in setting **\$1** to **-**.

Using **+** rather than **-** causes these flags to be turned off. These flags can also be used upon invocation of the shell. The current set of flags may be found in **\$-**. The remaining arguments are positional parameters and are assigned, in order, to **\$1**, **\$2**, If no arguments are given the values of all names are printed.

shift [*n*]

The positional parameters from **\$n+1** ... are renamed **\$1** If *n* is not given, it is assumed to be 1.

test

Evaluate conditional expressions. See *test(1)* for usage and description.

times

Print the accumulated user and system times for processes run from

the shell.

trap [*arg*] [*n*] ...

The command *arg* is to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If *arg* is absent all trap(s) *n* are reset to their original values. If *arg* is the null string this signal is ignored by the shell and by the commands it invokes. If *n* is 0 the command *arg* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

type [*name* ...]

For each *name*, indicate how it would be interpreted if used as a command name.

ulimit [*n*]

Impose a size limit of *n* blocks on files written by the shell and its child processes (files of any size may be read). If *n* is omitted, the current limit is printed. You may lower your own ulimit, but only a super-user (see *su(1M)*) can raise a ulimit.

umask [*nnn*]

The user file-creation mask is set to *nnn* (see *umask(1)*). If *nnn* is omitted, the current value of the mask is printed.

unset [*name* ...]

For each *name*, remove the corresponding variable or function. The variables **PATH**, **PS1**, **PS2**, **MAILCHECK** and **IFS** cannot be unset.

wait [*n*]

Wait for your background process whose process id is *n* and report its termination status. If *n* is omitted, all your shell's currently active background processes are waited for and the return code will be zero.

Invocation

If the shell is invoked through *exec(2)* and the first character of argument zero is **-**, commands are initially read from */etc/profile* and from *\$HOME/.profile*, if such files exist. Thereafter, commands are read as described below, which is also the case when the shell is invoked as */bin/sh*. The flags below are interpreted by the shell on invocation only; Note that unless the **-c** or **-s** flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file:

-c *string* If the **-c** flag is present commands are read from *string*.

-s If the **-s** flag is present or if no arguments remain commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output (except for *Special Commands*) is written to file descriptor 2.

- i If the `—i` flag is present or if the shell input and output are attached to a terminal, this shell is *interactive*. In this case `TERMINATE` is ignored (so that `kill 0` does not kill an interactive shell) and `INTERRUPT` is caught and ignored (so that `wait` is interruptible). In all cases, `QUIT` is ignored by the shell.
- r If the `—r` flag is present the shell is a restricted shell.

The remaining flags and arguments are described under the `set` command above.

rsh Only

`rsh` is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of `rsh` are identical to those of `sh`, except that the following are disallowed:

- changing directory (see `cd(1)`),
- setting the value of `$PATH`,
- specifying path or command names containing `/`,
- redirecting output (`>` and `>>`).

The restrictions above are enforced after `.profile` is interpreted.

A restricted shell can be invoked in one of the following ways: (1) `rsh` is the file name part of the last entry in the `/etc/passwd` file (see `passwd(4)`); (2) the environment variable `SHELL` exists and `rsh` is the file name part of its value; (3) the shell is invoked and `rsh` is the file name part of argument 0; (4) the shell is invoked with the `—r` option.

When a command to be executed is found to be a shell procedure, `rsh` invokes `sh` to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the `.profile` (see `profile(4)`) has complete control over user actions by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (i.e., `/usr/rbin`) that can be safely invoked by a restricted shell. Some systems also provide a restricted editor, `red`.

EXIT STATUS

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the `exit` command above).

FILES

`/etc/profile`
`$HOME/.profile`
`/tmp/sh*`
`/dev/null`

SEE ALSO

cd(1), echo(1), env(1), getopts(1), intro(1), login(1), newgrp(1), pwd(1), test(1), umask(1), wait(1).
dup(2), exec(2), fork(2), pipe(2), signal(2), ulimit(2) in the *Programmer's Reference Manual*.
profile(4) in the *System Administrator's Reference Manual*.

CAVEATS

Words used for filenames in input/output redirection are not interpreted for filename generation (see "File Name Generation," above). For example, **cat file1 >a*** will create a file named **a***.

Because commands in pipelines are run as separate processes, variables set in a pipeline have no effect on the parent shell.

If you get the error message *cannot fork, too many processes*, try using the *wait(1)* command to clean up your background processes. If this doesn't help, the system process table is probably full or you have too many active foreground processes. (There is a limit to the number of process ids associated with your login, and to the number the system can keep track of.)

BUGS

If a command is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to *exec* the original command. Use the **hash** command to correct this situation.

If you move the current directory or one above it, **pwd** may not give the correct response. Use the **cd** command with a full path name to correct this situation.

Not all the processes of a 3- or more-stage pipeline are children of the shell, and thus cannot be waited for.

For *wait n*, if *n* is not an active process id, all your shell's currently active background processes are waited for and the return code will be zero.



NAME

shl – shell layer manager

SYNOPSIS

shl

DESCRIPTION

shl allows a user to interact with more than one shell from a single terminal. The user controls these shells, known as *layers*, using the commands described below.

The *current layer* is the layer which can receive input from the keyboard. Other layers attempting to read from the keyboard are blocked. Output from multiple layers is multiplexed onto the terminal. To have the output of a layer blocked when it is not current, the *stty* option **loblk** may be set within the layer.

The *stty* character **switch** (set to ^Z if NUL) is used to switch control to *shl* from a layer. *shl* has its own prompt, **>>>**, to help distinguish it from a layer.

A *layer* is a shell which has been bound to a virtual tty device (*/dev/sxt???*). The virtual device can be manipulated like a real tty device using *stty*(1) and *ioctl*(2). Each layer has its own process group id.

Definitions

A *name* is a sequence of characters delimited by a blank, tab or new-line. Only the first eight characters are significant. The *names* (1) through (7) cannot be used when creating a layer. They are used by *shl* when no name is supplied. They may be abbreviated to just the digit.

Commands

The following commands may be issued from the *shl* prompt level. Any unique prefix is accepted.

create [*name*]

Create a layer called *name* and make it the current layer. If no argument is given, a layer will be created with a name of the form (#) where # is the last digit of the virtual device bound to the layer. The shell prompt variable **PS1** is set to the name of the layer followed by a space. A maximum of seven layers can be created.

block *name* [*name* ...]

For each *name*, block the output of the corresponding layer when it is not the current layer. This is equivalent to setting the *stty* option **-loblk** within the layer.

delete *name* [*name* ...]

For each *name*, delete the corresponding layer. All processes in the process group of the layer are sent the **SIGHUP** signal (see *signal*(2)).

help (or ?)

Print the syntax of the *shl* commands.

layers [**-l**] [*name* ...]

For each *name*, list the layer name and its process group. The **-l** option produces a *ps*(1)-like listing. If no arguments are given, information is presented for all existing layers.

resume [*name*]

Make the layer referenced by *name* the current layer. If no argument is given, the last existing current layer will be resumed.

toggle Resume the layer that was current before the last current layer.**unblock** *name* [*name* ...]

For each *name*, do not block the output of the corresponding layer when it is not the current layer. This is equivalent to setting the *stty* option **-loblk** within the layer.

quit Exit *shl*. All layers are sent the SIGHUP signal.

name Make the layer referenced by *name* the current layer.

FILES

/dev/sxt???	Virtual tty devices
\$SHELL	Variable containing path name of the shell to use (default is /bin/sh).

SEE ALSO

sh(1), stty(1).

ioctl(2), signal(2) in the *Programmer's Reference Manual*.

sxt(7) in the *System Administrator's Reference Manual*.

NAME

`sleep` – suspend execution for an interval

SYNOPSIS

`sleep time`

DESCRIPTION

`sleep` suspends execution for *time* seconds. It is used to execute a command after a certain amount of time, as in:

```
(sleep 105; command)&
```

or to execute a command every so often, as in:

```
while true
do
    command
    sleep 37
done
```

SEE ALSO

`alarm(2)`, `sleep(3C)` in the *Programmer's Reference Manual*.



NAME

sort – sort and/or merge files

SYNOPSIS

sort [**-cmu**] [**-ooutput**] [**-ykmem**] [**-zrecsz**] [**-dfiMnr**] [**-btx**]
 [**+pos1** [**-pos2**]] [**files**]

DESCRIPTION

sort sorts lines of all the named files together and writes the result on the standard output. The standard input is read if **-** is used as a file name or no input files are named.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line, and ordering is lexicographic by bytes in machine collating sequence.

The following options alter the default behavior:

- c** Check that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.
- m** Merge only, the input files are already sorted.
- u** Unique: suppress all but one in each set of lines having equal keys.

-ooutput

The argument given is the name of an output file to use instead of the standard output. This file may be the same as one of the inputs. There may be optional blanks between **-o** and *output*.

-ykmem

The amount of main memory used by the sort has a large impact on its performance. Sorting a small file in a large amount of memory is a waste. If this option is omitted, *sort* begins using a system default memory size, and continues to use more space as needed. If this option is presented with a value, *kmem*, *sort* will start using that number of kilobytes of memory, unless the administrative minimum or maximum is violated, in which case the corresponding extremum will be used. Thus, **-y0** is guaranteed to start with minimum memory. By convention, **-y** (with no argument) starts with maximum memory.

-zrecsz

The size of the longest line read is recorded in the sort phase so buffers can be allocated during the merge phase. If the sort phase is omitted via the **-c** or **-m** options, a popular system default size will be used. Lines longer than the buffer size will cause *sort* to terminate abnormally. Supplying the actual number of bytes in the longest line to be merged (or some larger value) will prevent abnormal termination.

The following options override the default ordering rules.

- d** "Dictionary" order: only letters, digits, and blanks (spaces and tabs) are significant in comparisons.
- f** Fold lower-case letters into upper case.
- i** Ignore non-printable characters.

- M Compare as months. The first three non-blank characters of the field are folded to upper case and compared. For example, in English the sorting order is "JAN" < "FEB" < ... < "DEC". Invalid fields compare low to "JAN". The -M option implies the -b option (see below).
- n An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. The -n option implies the -b option (see below). Note that the -b option is only effective when restricted sort key specifications are in effect.
- r Reverse the sense of comparisons.

When ordering options appear before restricted sort key specifications, the requested ordering rules are applied globally to all sort keys. When attached to a specific sort key (described below), the specified ordering options override all global ordering options for that key.

The notation $+pos1 -pos2$ restricts a sort key to one beginning at $pos1$ and ending just before $pos2$. The characters at position $pos1$ and just before $pos2$ are included in the sort key (provided that $pos2$ does not precede $pos1$). A missing $-pos2$ means the end of the line.

Specifying $pos1$ and $pos2$ involves the notion of a field, a minimal sequence of characters followed by a field separator or a new-line. By default, the first blank (space or tab) of a sequence of blanks acts as the field separator. All blanks in a sequence of blanks are considered to be part of the next field; for example, all blanks at the beginning of a line are considered to be part of the first field. The treatment of field separators can be altered using the options:

- b Ignore leading blanks when determining the starting and ending positions of a restricted sort key. If the -b option is specified before the first $+pos1$ argument, it will be applied to all $+pos1$ arguments. Otherwise, the **b** flag may be attached independently to each $+pos1$ or $-pos1$ argument (see below).
- tx Use x as the field separator character; x is not considered to be part of a field (although it may be included in a sort key). Each occurrence of x is significant (for example, xx delimits an empty field).

$pos1$ and $pos2$ each have the form $m.n$ optionally followed by one or more of the flags **bdfinr**. A starting position specified by $+m.n$ is interpreted to mean the $n+1$ st character in the $m+1$ st field. A missing $.n$ means $.0$, indicating the first character of the $m+1$ st field. If the **b** flag is in effect n is counted from the first non-blank in the $m+1$ st field; $+m.0b$ refers to the first non-blank character in the $m+1$ st field.

A last position specified by $-m.n$ is interpreted to mean the n th character (including separators) after the last character of the m th field. A missing $.n$ means $.0$, indicating the last character of the m th field. If the **b** flag is in effect n is counted from the last leading blank in the $m+1$ st field; $-m.1b$ refers to the first non-blank in the $m+1$ st field.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant.

EXAMPLES

Sort the contents of *infile* with the second field as the sort key:

```
sort +1 -2 infile
```

Sort, in reverse order, the contents of *infile1* and *infile2*, placing the output in *outfile* and using the first character of the second field as the sort key:

```
sort -r -o outfile +1.0 -1.2 infile1 infile2
```

Sort, in reverse order, the contents of *infile1* and *infile2* using the first non-blank character of the second field as the sort key:

```
sort -r +1.0b -1.1b infile1 infile2
```

Print the password file (*passwd*(4)) sorted by the numeric user ID (the third colon-separated field):

```
sort -t: +2n -3 /etc/passwd
```

Print the lines of the already sorted file *infile*, suppressing all but the first occurrence of lines having the same third field (the options `-um` with just one input file make the choice of a unique representative from a set of equal lines predictable):

```
sort -um +2 -3 infile
```

FILES

/usr/tmp/stm???

SEE ALSO

`comm(1)`, `join(1)`, `uniq(1)`.

WARNINGS

Comments and exits with non-zero status for various trouble conditions (for example, when input lines are too long), and for disorder discovered under the `-c` option. When the last line of an input file is missing a **new-line** character, *sort* appends one, prints a warning message, and continues.

sort does not guarantee preservation of relative line ordering on equal keys.



NAME

`spell`, `hashmake`, `spellin`, `hashcheck` – find spelling errors

SYNOPSIS

```
spell [ -v ] [ -b ] [ -x ] [ -l ] [ +local_file ] [ files ]
/usr/lib/spell/hashmake
/usr/lib/spell/spellin n
/usr/lib/spell/hashcheck spelling_list
```

DESCRIPTION

`spell` collects words from the named *files* and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes, and/or suffixes) from words in the spelling list are printed on the standard output. If no *files* are named, words are collected from the standard input.

`spell` ignores most `troff(1)`, `tbl(1)`, and `eqn(1)` constructions.

Under the `-v` option, all words not literally in the spelling list are printed, and plausible derivations from the words in the spelling list are indicated.

Under the `-b` option, British spelling is checked. Besides preferring *centre*, *colour*, *programme*, *speciality*, *travelled*, etc., this option insists upon *-ise* in words like *standardise*, Fowler and the OED to the contrary notwithstanding.

Under the `-x` option, every plausible stem is printed with `=` for each word.

By default, `spell` (like `deroff(1)`) follows chains of included files (`.so` and `.nx` `troff(1)` requests), unless the names of such included files begin with `/usr/lib`. Under the `-l` option, `spell` will follow the chains of *all* included files.

Under the `+local_file` option, words found in *local_file* are removed from `spell`'s output. *Local_file* is the name of a user-provided file that contains a sorted list of words, one per line. With this option, the user can specify a set of words that are correct spellings (in addition to `spell`'s own spelling list) for each job.

The spelling list is based on many sources, and while more haphazard than an ordinary dictionary, is also more effective with respect to proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine, and chemistry is light.

Pertinent auxiliary files may be specified by name arguments, indicated below with their default settings (see *FILES*). Copies of all output are accumulated in the history file. The stop list filters out misspellings (e.g., `thier=thy+y+ier`) that would otherwise pass.

Three routines help maintain and check the hash lists used by `spell`:

hashmake Reads a list of words from the standard input and writes the corresponding nine-digit hash code on the standard output.

spellin Reads *n* hash codes from the standard input and writes a compressed spelling list on the standard output.

hashcheck Reads a compressed *spelling_list* and recreates the nine-digit hash codes for all the words in it; it writes these codes on the standard output.

FILES

D_SPELL=/usr/lib/spell/hlist[ab]	hashed spelling lists, American & British
S_SPELL=/usr/lib/spell/hstop	hashed stop list
H_SPELL=/usr/lib/spell/spellhist	history file
/usr/lib/spell/spellprog	program

SEE ALSO

deroff(1), sed(1), sort(1), tee(1).
 eqn(1), tbl(1), troff(1) in the *DOCUMENTER'S WORKBENCH Software 2.0 Technical Discussion and Reference Manual*.

BUGS

The spelling list's coverage is uneven; new installations will probably wish to monitor the output for several months to gather local additions; typically, these are kept in a separate local file that is added to the hashed *spelling_list* via *spellin*.

NAME

`split` – split a file into pieces

SYNOPSIS

`split` [*-n*] [*file* [*name*]]

DESCRIPTION

split reads *file* and writes it in *n*-line pieces (default 1000 lines) onto a set of output files. The name of the first output file is *name* with **aa** appended, and so on lexicographically, up to **zz** (a maximum of 676 files). *Name* cannot be longer than 12 characters. If no output name is given, **x** is default.

If no input file is given, or if **-** is given in its stead, then the standard input file is used.

SEE ALSO

`bfs(1)`, `csplit(1)`.



NAME

stty – set the options for a terminal

SYNOPSIS

stty [**-a**] [**-g**] [options]

DESCRIPTION

stty sets certain terminal I/O options for the device that is the current standard input; without arguments, it reports the settings of certain options.

In this report, if a character is preceded by a caret (^), then the value of that option is the corresponding CTRL character (e.g., “^h” is CTRL-h ; in this case, recall that CTRL-h is the same as the “back-space” key.) The sequence “” means that an option has a null value. For example, normally **stty -a** will report that the value of **switch** is “”; however, if **shl** (1) or **layers** (1) has been invoked, **stty -a** will have the value “^z”.

-a reports all of the option settings;

-g reports current settings in a form that can be used as an argument to another *stty* command.

Options in the last group are implemented using options in the previous groups. Note that many combinations of options make no sense, but no sanity checking is performed. The options are selected from the following:

Control Modes

parenb (**-parenb**) enable (disable) parity generation and detection.

parodd (**-parodd**) select odd (even) parity.

cs5 cs6 cs7 cs8 select character size (see *termio*(7)).

0 hang up phone line immediately.

110 300 600 1200 1800 2400 4800 9600 19200 38400

Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)

hupcl (**-hupcl**) hang up (do not hang up) Dataphone connection on last close.

hup (**-hup**) same as **hupcl** (**-hupcl**).

cstopb (**-cstopb**) use two (one) stop bits per character.

cread (**-cread**) enable (disable) the receiver.

local (**-local**) n assume a line without (with) modem control.

loblk (**-loblk**) block (do not block) output from a non-current layer.

Input Modes

ignbrk (**-ignbrk**) ignore (do not ignore) break on input.

brkint (**-brkint**) signal (do not signal) INTR on break.

ignpar (**-ignpar**) ignore (do not ignore) parity errors.

parmrk (**-parmrk**) mark (do not mark) parity errors (see *termio*(7)).

inpck (**-inpck**) enable (disable) input parity checking.

istrip (**-istrip**) strip (do not strip) input characters to seven bits.

inlcr (**-inlcr**) map (do not map) NL to CR on input.

igncr (**-igncr**) ignore (do not ignore) CR on input.

icrnl (-icrnl)	map (do not map) CR to NL on input.
iuclic (-iuclic)	map (do not map) upper-case alphabets to lower case on input.
ixon (-ixon)	enable (disable) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.
ixany (-ixany)	allow any character (only DC1) to restart output.
ixoff (-ixoff)	request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

Output Modes

opost (-opost)	post-process output (do not post-process output; ignore all other output modes).
olcuc (-olcuc)	map (do not map) lower-case alphabets to upper case on output.
onlcr (-onlcr)	map (do not map) NL to CR-NL on output.
ocrnl (-ocrnl)	map (do not map) CR to NL on output.
onocr (-onocr)	do not (do) output CRs at column zero.
onlret (-onlret)	on the terminal NL performs (does not perform) the CR function.
ofill (-ofill)	use fill characters (use timing) for delays.
ofdel (-ofdel)	fill characters are DELs (NULs).
cr0 cr1 cr2 cr3	select style of delay for carriage returns (see <i>termio(7)</i>).
nl0 nl1	select style of delay for line-feeds (see <i>termio(7)</i>).
tab0 tab1 tab2 tab3	select style of delay for horizontal tabs (see <i>termio(7)</i>).
bs0 bs1	select style of delay for backspaces (see <i>termio(7)</i>).
ff0 ff1	select style of delay for form-feeds (see <i>termio(7)</i>).
vt0 vt1	select style of delay for vertical tabs (see <i>termio(7)</i>).

Local Modes

isig (-isig)	enable (disable) the checking of characters against the special control characters INTR, QUIT, and SWTCH.
icanon (-icanon)	enable (disable) canonical input (ERASE and KILL processing).
xcase (-xcase)	canonical (unprocessed) upper/lower-case presentation.
echo (-echo)	echo back (do not echo back) every character typed.
echoe (-echoe)	echo (do not echo) ERASE character as a backspace-space-backspace string. Note: this mode will erase the ERASEed character on many CRT terminals; however, it does <i>not</i> keep track of column position and, as a result, may be confusing on escaped characters, tabs, and backspaces.
echok (-echok)	echo (do not echo) NL after KILL character.
lfkc (-lfkc)	the same as echok (-echok); obsolete.
echonl (-echonl)	echo (do not echo) NL.
noflsh (-noflsh)	disable (enable) flush after INTR, QUIT, or SWTCH.
stwrap (-stwrap)	disable (enable) truncation of lines longer than 79 characters on a synchronous line. (Does not apply to the 3B2.)

stflush (**-stflush**) enable (disable) flush on a synchronous line after every *write*(2). (Does not apply to *stappl*.)

stappl (**-stappl**) use application mode (use line mode) on a synchronous line. (Does not apply to the 3B2.)

Control Assignments

control-character c

set *control-character* to *c*, where *control-character* is **erase**, **kill**, **intr**, **quit**, **swtch**, **eof**, **ctab**, **min**, or **time** (**ctab** is used with **-stappl**; **min** and **time** are used with **-icanon**; see *termio*(7)). If *c* is preceded by an (escaped from the shell) caret (^), then the value used is the corresponding CTRL character (e.g., "**^d**" is a CTRL-d); "**^?"** is interpreted as DEL and "**^-**" is interpreted as undefined.

line i set line discipline to *i* ($0 < i < 127$).

Combination Modes

evenp or **parity**

enable **parenb** and **cs7**.

oddp

enable **parenb**, **cs7**, and **parodd**.

-parity, **-evenp**, or **-oddp**

disable **parenb**, and set **cs8**.

raw (**-raw** or **cooked**)

enable (disable) raw input and output (no ERASE, KILL, INTR, QUIT, SWTCH, EOT, or output post processing).

nl (**-nl**)

unset (set) **icrnl**, **onlcr**. In addition **-nl** unsets **inlcr**, **igncr**, **ocrnl**, and **onlret**.

lcase (**-lcase**)

set (unset) **xcase**, **iuclc**, and **olcuc**.

LCASE (**-LCASE**)

same as **lcase** (**-lcase**).

tabs (**-tabs** or **tab3**)

preserve (expand to spaces) tabs when printing.

ek

reset ERASE and KILL characters back to normal # and @.

sane

resets all modes to some reasonable values.

term

set all modes suitable for the terminal type *term*, where *term* is one of **tty33**, **tty37**, **vt05**, **tn300**, **ti700**, or **tek**.

SEE ALSO

tabs(1).

ioctl(2) in the *Programmer's Reference Manual*.

termio(7) in the *System Administrator's Reference Manual*.



NAME

`su` — become super-user or another user

SYNOPSIS

`su [-] [name [arg ...]]`

DESCRIPTION

`su` allows one to become another user without logging off. The default user *name* is **root** (i.e., super-user).

To use `su`, the appropriate password must be supplied (unless one is already **root**). If the password is correct, `su` will execute a new shell with the real and effective user ID set to that of the specified user. The new shell will be the optional program named in the shell field of the specified user's password file entry (see `passwd(4)`), or `/bin/sh` if none is specified (see `sh(1)`). To restore normal user ID privileges, type an EOF (`cntrl-d`) to the new shell.

Any additional arguments given on the command line are passed to the program invoked as the shell. When using programs like `sh(1)`, an *arg* of the form `-c string` executes *string* via the shell and an *arg* of `-r` will give the user a restricted shell.

The following statements are true only if the optional program named in the shell field of the specified user's password file entry is like `sh(1)`. If the first argument to `su` is a `-`, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is `-`, thus causing first the system's profile (`/etc/profile`) and then the specified user's profile (`.profile` in the new HOME directory) to be executed. Otherwise, the environment is passed along with the possible exception of `$PATH`, which is set to `/bin:/etc:/usr/bin` for **root**. Note that if the optional program used as the shell is `/bin/sh`, the user's `.profile` can check *arg0* for `-sh` or `-su` to determine if it was invoked by `login(1)` or `su(1)`, respectively. If the user's program is other than `/bin/sh`, then `.profile` is invoked with an *arg0* of `-program` by both `login(1)` and `su(1)`.

All attempts to become another user using `su` are logged in the log file `/usr/adm/sulog`.

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute:

```
su bin
```

To become user **bin** but change the environment to what would be expected if **bin** had originally logged in, execute:

```
su - bin
```

To execute *command* with the temporary environment and permissions of user **bin**, type:

```
su - bin -c "command args"
```

FILES

/etc/passwd	system's password file
/etc/profile	system's profile
\$HOME/.profile	user's profile
/usr/adm/sulog	log file

SEE ALSO

env(1), login(1), sh(1) in the *User's Reference Manual*.
passwd(4), profile(4), environ(5) in the *System Administrator's Reference Manual*.

NAME

`sum` – print checksum and block count of a file

SYNOPSIS

`sum [-r] file`

DESCRIPTION

sum calculates and prints a 16-bit checksum for the named file, and also prints the number of blocks in the file. It is typically used to look for bad spots, or to validate a file communicated over some transmission line. The option `-r` causes an alternate algorithm to be used in computing the checksum.

SEE ALSO

`wc(1)`.

DIAGNOSTICS

“Read error” is indistinguishable from end of file on most devices; check the block count.



NAME

sync – update the super block

SYNOPSIS

sync

DESCRIPTION

sync executes the *sync* system primitive. If the system is to be stopped, *sync* must be called to insure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See *sync(2)* for details.

NOTE

If you have done a write to a file on a remote machine in a Remote File Sharing environment, you cannot use *sync* to force buffers to be written out to disk on the remote machine. *sync* will only write local buffers to local disks.

SEE ALSO

sync(2) in the *Programmer's Reference Manual*.



NAME

`sysadm` – menu interface to do system administration

SYNOPSIS

`sysadm` [*sub-command*]

DESCRIPTION

This command, when invoked without an argument, presents a menu of system administration sub-commands, from which the user selects. If the optional argument is presented, the named sub-command is run or the named sub-menu is presented.

The *sysadm* command may be given a password. See `admpasswd` in the SUB-COMMANDS section.

SUB-COMMANDS

The following menus of sub-commands are available. (The number of bullets (•) in front of each item indicates the level of the menu or subcommand.)

- diagnostics

- system diagnostics menu

- These subcommands look for and sometimes repair problems in the system. Those subcommands that issue reports allow you to determine if there are detectable problems. Commands that attempt repair are for repair people only. You must know what you are doing!

- diskrepair

- advice on repair of built-in disk errors

- This subcommand advises you on how to go about repairing errors that occur on built-in disks.

- WARNING:** Because this is a repair function, it should only be performed by qualified service personnel.

- NOTE:** Reports of disk errors most probably result in the loss of files and/or damage to data. It will be necessary to restore the repaired disk from backup copies.

- diskreport

- report on built-in disk errors

- This subcommand shows you if the system has collected any information indicating that there have been errors while reading the built-in disks. You can request either summary or full reports. The summary report provides sufficient information about disk errors to determine if repair should be attempted. If the message no errors logged is part of the report, then there is probably no damage. If a number of errors is reported, there is damage and you should call for service. The full report gives additional detail for the expert repair person trouble shooting complicated problems.

NOTE: Reports of disk errors most probably result in the loss of files and/or damage to data. It will be necessary to restore the repaired disk from backup copies.

- diskmgmt
disk management menu

The subcommands in this menu provide functions for using removable disks. The subcommands include the ability to format disks, copy disks, and to use disks as mountable file systems. It also contains a menu of subcommands for handling non-removable media.

- checkfsys
check a removable disk file system for errors

Checkfsys checks a file system on a removable disk for errors. If there are errors, this procedure attempts to repair them.

- cpdisk
make exact copies of a removable disk

This procedure copies the contents of a removable disk into the machine and then allows the user to make exact copies of it. These copies are identical to the original in every way. The copies are made by first reading the original removable disk entirely into the machine and then writing it out onto duplicate disks. The procedure will fail if there is not enough space in the system to hold the original disk.

- erase
erase data from removable disk

This procedure erases a removable disk by overwriting it with null bytes. The main purpose is to remove data that the user does not want seen. Once performed, this operation is irreversible.

- format
format new removable disks

Format prepares new removable disks for use. Once formatted, programs and data can be written on the disks.

- harddisk
hard disk management menu

The subcommands in this menu provide functions for using hard disks. For each hard disk, the disk can be partitioned with default partitioning or the current disk partitioning can be displayed.

- display
display hard disk partitioning

Display will allow the user to display the hard disk partitioning. This will inform the user of current disk partitioning information.

- partitioning
 - partition a hard disk

Partitioning configures hard disks. This will allow you to partition a hard disk according to the default partitioning.

- rmdisk
 - remove a hard disk

Removes a hard disk from the system configuration. It may then be physically disconnected (once the machine has been turned off) or freshly partitioned (after the machine has been restarted).

- makefsys
 - create a new file system on a removable disk

Makefsys creates a new file system on a removable disk which can then store data which the user does not wish to keep on the hard disk. When "mounted", the file system has all the properties of a file kept on the hard disk, except that it is smaller.

- mountfsys
 - mount a removable disk file system

Mountfsys mounts a file system, found on a removable disk, making it available to the user. The file system is unmounted with the "umountfsys" command. **THE DISK MUST NOT BE REMOVED WHILE THE FILE SYSTEM IS STILL MOUNTED. IF THE FILE SYSTEM HAS BEEN MOUNTED WITH THE mountfsys COMMAND, IT MUST BE UNMOUNTED WITH umountfsys.**

- umountfsys
 - unmount a removable disk file system

Umountfsys unmounts a file system, allowing the user to remove the disk. **THE DISK MUST NOT BE REMOVED UNTIL THE FILE SYSTEM IS UNMOUNTED. umountfsys MAY ONLY BE USED TO UNMOUNT FILE SYSTEMS MOUNTED WITH THE mountfsys COMMAND.**

- filegmt
 - file management menu

The subcommands in this menu allow the user to protect files on the hard disk file systems by copying them onto diskettes and later restoring them to the hard disk by copying them back. Subcommands are also provided to determine which files might be best kept on diskette based on age or size.

- backup

backup files from integral hard disk to removable disk or tape

Backup saves copies of files from the integral hard disk file systems to removable disk or tape. There are two kinds of backups:

COMPLETE – copies all files (useful in case of serious file system damage)

INCREMENTAL – copies files changed since the last backup

The normal usage is to do a complete backup of each file system and then periodically do incremental backups. Two cycles are recommended (one set of complete backups and several incrementals to each cycle). Files backed up with "backup" are restored using "restore".

- bupsched

backup reminder scheduling menu

Backup scheduling is used to schedule backup reminder messages and backup reminder checks. Backup reminder messages are sent to the console to remind the administrator to backup particular file systems when the machine is shutdown or a reminder check has been run during the specified time period.

Backup reminder checks specify particular times at which the system will check to see if any backup reminder messages have been scheduled.

- schedcheck

schedule backup reminder checks

Backup reminder checks are run at specific times to check to see if any reminders are scheduled. The user specifies the times at which the check is to be run. Checks are run for the reminder messages scheduled by *schedmsg*.

- schedmsg

schedule backup reminder message

Backup reminder messages are sent to the console if the machine is shutdown or a reminder check has been scheduled. The user specifies the times at which it is appropriate to send a message and the file systems to be included in the message.

- diskuse

display how much of the hard disk is being used

Diskuse lets the user know what percentage of the hard disk is currently occupied by files. The list is organized by file system names.

- fileage
 - list files older than a particular date

 - Fileage prints the names of all files older than the date specified by the user. If no date is entered, all files older than 90 days will be listed.
- filesize
 - list the largest files in a particular directory

 - Filesize prints the names of the largest files in a specific directory. If no directory is specified, the `/usr/admin` directory will be used. If the user does not specify how many large files to list, 10 files will be listed.
- restore
 - restore files from "backup" and "store" media to integral hard disk

 - Restore copies files from disks and tapes made by "backup" and "store" back onto the hard disk. You can restore individual files, directories of files, or the entire contents of a disk or tape. The user can restore from both "incremental" and "complete" media. The user can also list the names of files stored on the disk or tape.
- store
 - store files and directories of files onto disk or tape

 - Store copies files from the integral hard disk to disk or tape and allows the user to optionally verify that they worked and to optionally remove them when done. Typically, these would be files that the user wants to archive or restrict access to. The user can store single files and directories of files. Use the "restore" command to put stored files back on the integral hard disk and to list the files stored.
- machinemgmt
 - machine management menu

 - Machine management functions are tools used to operate the machine, e.g., turn it off, reboot, or go to the firmware monitor.
- autold
 - set automatic boot device, default manual boot program

 - This procedure specifies the default manual program to boot from firmware and/or the device to be used when automatically rebooting.

- firmware

stop all running programs then enter firmware mode

This procedure will stop all running programs, close any open files, write out information to the disk (such as directory information), then enter the firmware mode. (Machine diagnostics and other special functions that are not available on the UNIX system.)

- floppykey

create a "floppy key" removable disk

The "floppy key" removable disk allows the user to enter firmware mode if the firmware password has been changed and then forgotten. Thus the "floppy key" is just that, the "key" to the system and should be protected as such.

- powerdown

stop all running programs, then turn off the machine

Powerdown will stop all running programs, close any open files, write out information to disk (such as directory information), then turn the machine power off.

- reboot

stop all running programs then reboot the machine

Reboot will stop all running programs, close any open files, write out information to disk (such as directory information), then reboot the machine. This can be used to get out of some types of system trouble, such as when a process cannot be killed.

- whoson

print list of users currently logged onto the system

Whoson prints the login ID, terminal device number, and sign-on time of all users who are currently using the computer.

- packagemgmt

package management

These submenus and subcommands manage various software and hardware packages that you install on your machine. Not all optional packages add subcommands here.

- softwaremgmt
software management menu

These subcommands permit the user to install new software, remove software, and run software directly from the removable disk it is delivered on. The "remove" and "run" capabilities are dependent on the particular software packages. See the instructions delivered with each package.

- installpkg
install new software package onto integral hard disk

Install copies files from removable disk onto the integral hard disk and performs additional work if necessary so that the software can be run. From then on, the user will have access to those commands.

- listpkg
list packages already installed

This subcommand shows you a list of currently installed optional software packages.

- removepkg
remove previously installed package from integral hard disk

This subcommand displays a list of currently installed optional software packages. Actions necessary to remove the software packages specified by the user will then be performed. The removable disk used to "installpkg" the software is needed to remove it.

- runpkg
run software package without installing it

This package allows the user to run software from a removable disk without installing it permanently on the system. This is useful if the user does not use the software often or does not have enough room on the system. **WARNING:** Not all software packages have the ability to run their contents this way. See the instructions that come with the software package.

- syssetup
system setup menu

System setup routines allow the user to tell the computer what its environment looks like: what the date, time, and time zone is, what administration and system capabilities are to be under password control, what the machine's name is, etc. The first-time setup sequence is also here.

- **admpasswd**
assign or change administrative passwords

Admpasswd lets you set or make changes to passwords for administrative commands and logins such as setup and sysadm.

- **datetime**
set the date, time, time zone, and daylight savings time

Datetime tells the computer the date, time, time zone, and whether you observe Daylight Savings Time (DST). It is normally run once when the machine is first set up. If you observe DST, the computer will automatically start to observe it in the spring and return to Standard Time in the fall. The machine has to be turned off and turned back on again to guarantee that ALL times will be reported correctly. Most are correct the next time the user logs in.

- **nodename**
set the node name of this machine

This allows you to change the node name of this machine. The node name is used by various communications networks to identify this machine.

- **setup**
set up your machine the very first time

Setup allows the user to define the first login, to set the passwords on the user-definable administration logins and to set the time zone for your location.

- **syspasswd**
assign system passwords

Syspasswd lets the user set system passwords normally reserved for the very knowledgeable user. For this reason, this procedure may assign those passwords, but may not change or clear them. Once set, they may only be changed by the specific login or the "root" login.

- **ttymgmt**
terminal management

This procedure allows the user to manage the computer's terminal functions.

- **lineset**
show tty line settings and hunt sequences

The tty line settings are often hunt sequences where, if the first line setting does not work, the line "hunts" to the next line setting until one that does work comes by. This subcommand shows the various sequences with only specific line settings in them. It also shows each line setting in detail.

- **mklineset**
create new tty line settings and hunt sequences

This subcommand helps you to create tty line setting entries. You might want to add line settings that are not in the current set or create hunt sequences with only specific line settings in them. The created hunt sequences are circular; stepping past the last setting puts you on the first.

- **modtty**
show and optionally modify characteristics of tty lines

This subcommand reports and allows you to change the characteristics of tty lines (also called "ports").

- **usermgmt**
user management menu

These subcommands allow you to add, modify and delete the list of users that have access to your machine. You can also place them in separate groups so that they can share access to files within the group but protect themselves from other groups.

- **addgroup**
add a group to the system

Addgroup adds a new group name or ID to the computer. Group names and IDs are used to identify groups of users who desire common access to a set of files and directories.

- **adduser**
add a user to the system

Adduser installs a new login ID on the machine. You are asked a series of questions about the user and then the new entry is made. You can enter more than one user at a time. Once this procedure is finished, the new login ID is available.

- **delgroup**
delete a group from the system

Delgroup allows you to remove groups from the computer. The deleted group is no longer identified by name. However, files may still be identified with the group ID number.

- **deluser**
delete a user from the system

Deluser allows you to remove users from the computer. The deleted user's files are removed from the hard disk and their logins are removed from the `/etc/passwd` file.

- lsgroup
list groups in the system

Lsgroup will list all the groups that have been entered into the computer. This list is updated automatically by "addgroup" and "delgroup".

- lsuser
list users in the system

Lsuser will list all the users that have been entered into the computer. This list is updated automatically by "adduser" and "deluser".

- modadduser
modify defaults used by adduser

Modadduser allows the user to change some of the defaults used when adduser creates a new login. Changing the defaults does not effect any existing logins, only logins made from this point on.

- modgroup
make changes to a group on the system

Modgroup allows the user to change the name of a group that the user enters when "addgroup" is run to set up new groups.

- moduser
menu of commands to modify a user's login

This menu contains commands that modify the various aspects of a user's login.

- chgloginid
change a user's login ID

This procedure allows the user to change a user's login ID. Administrative and system logins cannot be changed.

- chgpasswd
change a user's passwd

This procedure allows removal or change of a user's password. Administrative and system login passwords cannot be changed. To change administrative and system login passwords, see the system setup menu: sysadm syssetup.

- chgshell
change a user's login shell

This procedure allows the user to change the command run when a user logs in. The login shell of the administrative and system logins cannot be changed by this procedure.

EXAMPLES

sysadm adduser

FILES

The files that support *sysadm* are found in **/usr/admin**.

The menu starts in directory **/usr/admin/menu**.



NAME

tabs - set tabs on a terminal

SYNOPSIS

tabs [tabspec] [-Ttype] [+mn]

DESCRIPTION

tabs sets the tab stops on the user's terminal according to the tab specification *tabspec*, after clearing any previous settings. The user's terminal must have remotely-settable hardware tabs.

tabspec Four types of tab specification are accepted for *tabspec*. They are described below: canned (*-code*), repetitive (*-n*), arbitrary (*n1,n2,...*), and file (*--file*). If no *tabspec* is given, the default value is **-8**, i.e., UNIX system "standard" tabs. The lowest column number is 1. Note that for *tabs*, column 1 always refers to the leftmost column on a terminal, even one whose column markers begin at 0, e.g., the DASI 300, DASI 300s, and DASI 450.

-code Use one of the codes listed below to select a *canned* set of tabs. The legal codes and their meanings are as follows:

- a** 1,10,16,36,72
Assembler, IBM S/370, first format
- a2** 1,10,16,40,72
Assembler, IBM S/370, second format
- c** 1,8,12,16,20,55
COBOL, normal format
- c2** 1,6,10,14,49
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a tab reaches column 12. Files using this tab setup should include a format specification as follows (see *fspec(4)*):
 <:t-c2 m6 s66 d:>
- c3** 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL compact format (columns 1-6 omitted), with more tabs than **-c2**. This is the recommended format for COBOL. The appropriate format specification is (see *fspec(4)*):
 <:t-c3 m6 s66 d:>
- f** 1,7,11,15,19,23
FORTRAN
- p** 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/I
- s** 1,10,55
SNOBOL
- u** 1,12,20,44
UNIVAC 1100 Assembler

- n** A *repetitive* specification requests tabs at columns $1+n$, $1+2*n$, etc. Of particular importance is the value **8**: this represents the UNIX system "standard" tab setting, and is the most likely tab setting to be found at a terminal. Another special case is the value **0**, implying no tabs at all.
- n1,n2,...** The *arbitrary* format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 40 numbers are allowed. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the formats **1,10,20,30**, and **1,10,+10,+10** are considered identical.
- file** If the name of a *file* is given, *tabs* reads the first line of the file, searching for a format specification (see *fspec(4)*). If it finds one there, it sets the tab stops according to it, otherwise it sets them as **-8**. This type of specification may be used to make sure that a tabbed file is printed with correct tab settings, and would be used with the *pr(1)* command:

tabs -- file; pr file

Any of the following also may be used; if a given flag occurs more than once, the last value given takes effect:

- Ttype** *tabs* usually needs to know the type of terminal in order to set tabs and always needs to know the type to set margins. *type* is a name listed in *term(5)*. If no **-T** flag is supplied, *tabs* uses the value of the environment variable **TERM**. If **TERM** is not defined in the *environment* (see *environ(5)*), *tabs* tries a sequence that will work for many terminals.
- +mn** The margin argument may be used for some terminals. It causes all tabs to be moved over *n* columns by making column $n+1$ the left margin. If **+m** is given without a value of *n*, the value assumed is **10**. For a TerminoNet, the first value in the tab list should be **1**, or the margin will move even further to the right. The normal (leftmost) margin on most terminals is obtained by **+m0**. The margin for most terminals is reset only when the **+m** flag is given explicitly.

Tab and margin setting is performed via the standard output.

EXAMPLES

- tabs -a** example using *-code* (*canned* specification) to set tabs to the settings required by the IBM assembler: columns 1, 10, 16, 36, 72.
- tabs -8** example of using *-n* (*repetitive* specification), where *n* is **8**, causes tabs to be set every eighth position:
1+(1*8), 1+(2*8), ... which evaluate to columns 9, 17, ...
- tabs 1,8,36** example of using *n1,n2,...* (*arbitrary* specification) to set tabs at columns 1, 8, and 36.

tabs **--\$HOME/fspec.list/att4425**

example of using **--file** (*file* specification) to indicate that tabs should be set according to the first line of *\$HOME/fspec.list/att4425* (see *fspec(4)*).

DIAGNOSTICS

<i>illegal tabs</i>	when arbitrary tabs are ordered incorrectly
<i>illegal increment</i>	when a zero or missing increment is found in an arbitrary specification
<i>unknown tab code</i>	when a <i>canned</i> code cannot be found
<i>can't open</i>	if --file option used, and file can't be opened
<i>file indirection</i>	if --file option used and the specification in that file points to yet another file. Indirection of this form is not permitted

SEE ALSO

newform(1), *pr(1)*, *tput(1)*.
fspec(4), *terminfo(4)*, *environ(5)*, *term(5)* in the *System Administrator's Reference Manual*.

NOTE

There is no consistency among different terminals regarding ways of clearing tabs and setting the left margin.

tabs clears only 20 tabs (on terminals requiring a long sequence), but is willing to set 64.

WARNING

The *tabspec* used with the *tabs* command is different from the one used with the *newform(1)* command. For example, **tabs -8** sets every eighth position; whereas **newform -i-8** indicates that tabs are set every eighth position.



NAME

`tail` – deliver the last part of a file

SYNOPSIS

`tail [±[number][lbc[f]]] [file]`

DESCRIPTION

`tail` copies the named file to the standard output beginning at a designated place. If no file is named, the standard input is used.

Copying begins at distance `+number` from the beginning, or `-number` from the end of the input (if `number` is null, the value 10 is assumed). `Number` is counted in units of lines, blocks, or characters, according to the appended option `l`, `b`, or `c`. When no units are specified, counting is by lines.

With the `-f` (“follow”) option, if the input file is not a pipe, the program will not terminate after the line of the input file has been copied, but will enter an endless loop, wherein it sleeps for a second and then attempts to read and copy further records from the input file. Thus it may be used to monitor the growth of a file that is being written by some other process. For example, the command:

```
tail -f fred
```

will print the last ten lines of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and killed. As another example, the command:

```
tail -15cf fred
```

will print the last 15 characters of the file `fred`, followed by any lines that are appended to `fred` between the time `tail` is initiated and killed.

SEE ALSO

`dd(1M)`.

BUGS

Tails relative to the end of the file are stored in a buffer, and thus are limited in length. Various kinds of anomalous behavior may happen with character special files.

WARNING

The `tail` command will only tail the last 4096 bytes of a file regardless of its line count.



NAME

tar — tape file archiver

SYNOPSIS

```
/etc/tar -c[vwfb[#s]] device block files ...
/etc/tar -r[vwb[#s]] device block [files ...]
/etc/tar -t[vf[#s]] device
/etc/tar -u[vwb[#s]] device block [files ...]
/etc/tar -x[lmowwf[#s]] device [files ...]
```

DESCRIPTION

tar saves and restores files on magnetic tape. Its actions are controlled by the *key* argument. The *key* is a string of characters containing one function letter (c, r, t, u, or x) and possibly followed by one or more function modifiers (v, w, f, b, and #). Other arguments to the command are *files* (or directory names) specifying which files are to be dumped or restored. In all cases, appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

The function portion of the key is specified by one of the following letters:

- r** Replace. The named *files* are written on the end of the tape. The c function implies this function.
- x** Extract. The named *files* are extracted from the tape. If a named file matches a directory whose contents had been written onto the tape, this directory is (recursively) extracted. Use the file or directory's relative path when appropriate, or *tar* will not find a match. The owner, modification time, and mode are restored (if possible). If no *files* argument is given, the entire content of the tape is extracted. Note that if several files with the same name are on the tape, the last one overwrites all earlier ones.
- t** Table. The names and other information for the specified files are listed each time that they occur on the tape. The listing is similar to the format produced by the *ls -l* command. If no *files* argument is given, all the names on the tape are listed.
- u** Update. The named *files* are added to the tape if they are not already there, or have been modified since last written on that tape. This key implies the r key.
- c** Create a new tape; writing begins at the beginning of the tape, instead of after the last file. This key implies the r key.

The characters below may be used in addition to the letter that selects the desired function. Use them in the order shown in the synopsis. **Note:** the only applicable device information for the 3B2 Computer is as follows:

/dev/mt/ctape [12...]

- #s** This modifier determines the drive on which the tape is mounted (replace # with the drive number) and the speed of the drive (replace s with l, m, or h for low, medium or high). The modifier tells *tar* to use a drive other than the default drive, or the drive specified with the -f option. For example, with the 5h modifier, *tar* would use /dev/mt/5h or /dev/mt0 instead of the default drives /dev/mt/0m

or `/dev/mt0`, respectively. However, if for example, "`-f /dev/rmt0 5h`" appeared on the command line, `tar` would use `/dev/rmt5h` or `/devmt0`. The default entry is `0m`.

- v** Verbose. Normally, `tar` does its work silently. The `v` (verbose) option causes it to type the name of each file it treats, preceded by the function letter. With the `t` function, `v` gives more information about the tape entries than just the name.
- w** What. This causes `tar` to print the action to be taken, followed by the name of the file, and then wait for the user's confirmation. If a word beginning with `y` is given, the action is performed. Any other input means "no". This is not valid with the `t` key.
- f** File. This causes `tar` to use the *device* argument as the name of the archive instead of `/dev/mt/0m` or `/dev/mt0`. If the name of the file is `-`, `tar` writes to the standard output or reads from the standard input, whichever is appropriate. Thus, `tar` can be used as the head or tail of a pipeline. `tar` can also be used to move hierarchies with the command:


```
cd fromdir; tar cf - . | (cd todir; tar xf -)
```
- b** Blocking Factor. This causes `tar` to use the *block* argument as the blocking factor for tape records. The default is 1, the maximum is 20. This function should not be supplied when operating on regular archives or block special devices. It is mandatory however, when reading archives on raw magnetic tape archives (see `f` above). The block size is determined automatically when reading tapes created on block special devices (key letters `x` and `t`).
- l** Link. This tells `tar` to complain if it cannot resolve all of the links to the files being dumped. If `l` is not specified, no error messages are printed.
- m** Modify. This tells `tar` to not restore the modification times. The modification time of the file will be the time of extraction.
- o** Ownership. This causes extracted files to take on the user and group identifier of the user running the program, rather than those on tape. This is only valid with the `x` key.

FILES

```
/dev/mt/*
/dev/mt*
/tmp/tar*
/dev/mt/ctape
/dev/mt/0m
/dev/rmt/0m
```

SEE ALSO

```
ar(1), cpio(1), ls(1).
```

DIAGNOSTICS

Complaints about bad key characters and tape read/write errors.
Complaints if enough memory is not available to hold the link tables.

BUGS

There is no way to ask for the n -th occurrence of a file.

Tape errors are handled ungracefully.

The **u** option can be slow.

The **b** option should not be used with archives that are going to be updated.

The current magnetic tape driver cannot backspace raw magnetic tape. If the archive is on a disk file, the **b** option should not be used at all, because updating an archive stored on disk can destroy it.

The current limit on file name length is 100 characters.

tar doesn't copy empty directories or special files.



NAME

tee – pipe fitting

SYNOPSIS

tee [**-i**] [**-a**] [file] ...

DESCRIPTION

tee transcribes the standard input to the standard output and makes copies in the *files*. The

-i ignore interrupts;

-a causes the output to be appended to the *files* rather than overwriting them.



NAME

`test` -- condition evaluation command

SYNOPSIS

```
test expr
[ expr ]
```

DESCRIPTION

`test` evaluates the expression *expr* and, if its value is true, sets a zero (true) exit status; otherwise, a non-zero (false) exit status is set; `test` also sets a non-zero exit status if there are no arguments. When permissions are tested, the effective user ID of the process is used.

All operators, flags, and brackets (brackets used as shown in the second SYNOPSIS line) must be separate arguments to the `test` command; normally these items are separated by spaces.

The following primitives are used to construct *expr*:

- `-r file` true if *file* exists and is readable.
- `-w file` true if *file* exists and is writable.
- `-x file` true if *file* exists and is executable.
- `-f file` true if *file* exists and is a regular file.
- `-d file` true if *file* exists and is a directory.
- `-c file` true if *file* exists and is a character special file.
- `-b file` true if *file* exists and is a block special file.
- `-p file` true if *file* exists and is a named pipe (fifo).
- `-u file` true if *file* exists and its set-user-ID bit is set.
- `-g file` true if *file* exists and its set-group-ID bit is set.
- `-k file` true if *file* exists and its sticky bit is set.
- `-s file` true if *file* exists and has a size greater than zero.
- `-t [fildev]` true if the open file whose file descriptor number is *fildev* (1 by default) is associated with a terminal device.
- `-z s1` true if the length of string *s1* is zero.
- `-n s1` true if the length of the string *s1* is non-zero.
- `s1 = s2` true if strings *s1* and *s2* are identical.
- `s1 != s2` true if strings *s1* and *s2* are *not* identical.
- `s1` true if *s1* is *not* the null string.
- `n1 -eq n2` true if the integers *n1* and *n2* are algebraically equal. Any of the comparisons `-ne`, `-gt`, `-ge`, `-lt`, and `-le` may be used in place of `-eq`.

These primaries may be combined with the following operators:

- ! unary negation operator.
- a binary *and* operator.
- o binary *or* operator (-a has higher precedence than -o).
- (expr) parentheses for grouping. Notice also that parentheses are meaningful to the shell and, therefore, must be quoted.

SEE ALSO

find(1), sh(1).

WARNING

If you test a file you own (the *-r*, *-w*, or *-x* tests), but the permission tested does not have the *owner* bit set, a non-zero (false) exit status will be returned even though the file may have the *group* or *other* bit set for that permission. The correct exit status will be set if you are super-user.

The = and != operators have a higher precedence than the -r through -n operators, and = and != always expect arguments; therefore, = and != cannot be used with the -r through -n operators.

If more than one argument follows the -r through -n operators, only the first argument is examined; the others are ignored, unless a -a or a -o is the second argument.

NAME

`time` — time a command

SYNOPSIS

`time` command

DESCRIPTION

The *command* is executed; after it is complete, *time* prints the elapsed time during the command, the time spent in the system, and the time spent in execution of the command. Times are reported in seconds.

The times are printed on standard error.

SEE ALSO

`times(2)` in the *Programmer's Reference Manual*.



NAME

`timex` – time a command; report process data and system activity

SYNOPSIS

`timex` [options] *command*

DESCRIPTION

The given *command* is executed; the elapsed time, user time and system time spent in execution are reported in seconds. Optionally, process accounting data for the *command* and all its children can be listed or summarized, and total system activity during the execution interval can be reported.

The output of *timex* is written on standard error.

Options are:

- p List process accounting records for *command* and all its children. This option works only if the process accounting software is installed. Suboptions *f*, *h*, *k*, *m*, *r*, and *t* modify the data items reported. The options are as follows:
 - f Print the *fork/exec* flag and system exit status columns in the output.
 - h Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This “hog factor” is computed as:
$$\frac{\text{(total CPU time)}}{\text{(elapsed time)}}$$
 - k Instead of memory size, show total kcore-minutes.
 - m Show mean core size (the default).
 - r Show CPU factor ($\text{user time}/(\text{system-time} + \text{user-time})$).
 - t Show separate system and user CPU times. The number of blocks read or written and the number of characters transferred are always reported.
- o Report the total number of blocks read or written and total characters transferred by *command* and all its children. This option works only if the process accounting software is installed.
- s Report total system activity (not just that due to *command*) that occurred during the execution interval of *command*. All the data items listed in *sar*(1) are reported.

SEE ALSO

sar(1).

WARNING

Process records associated with *command* are selected from the accounting file `/usr/adm/pacct` by inference, since process genealogy is not available. Background processes having the same user-id, terminal-id, and execution time window will be spuriously included.

EXAMPLES

A simple example:

```
timex -ops sleep 60
```

A terminal session of arbitrary complexity can be measured by timing a sub-shell:

```
timex -opskmt sh
      session commands
EOT
```

NAME

`touch` – update access and modification times of a file

SYNOPSIS

`touch [-amc] [mmddhhmm[yy]] files`

DESCRIPTION

`touch` causes the access and modification times of each argument to be updated. The file name is created if it does not exist. If no time is specified (see `date(1)`) the current time is used. The `-a` and `-m` options cause `touch` to update only the access or modification times respectively (default is `-am`). The `-c` option silently prevents `touch` from creating the file if it did not previously exist.

The return code from `touch` is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

SEE ALSO

`date(1)`.

`utime(2)` in the *Programmer's Reference Manual*.



NAME

tplot – graphics filters

SYNOPSIS

tplot [-Tterminal [-e raster]]

DESCRIPTION

These commands read plotting instructions (see *plot(4)*) from the standard input and in general produce, on the standard output, plotting instructions suitable for a particular *terminal*. If no *terminal* is specified, the environment parameter *\$TERM* (see *environ(5)*) is used. Known *terminals* are:

300 DASI 300.

300S DASI 300s.

450 DASI 450.

4014 Tektronix 4014.

ver Versatec D1200A. This version of *plot* places a scan-converted image in */usr/tmp/raster\$\$* and sends the result directly to the plotter device, rather than to the standard output. The *-e* option causes a previously scan-converted file *raster* to be sent to the plotter.

FILES

/usr/lib/t300

/usr/lib/t300s

/usr/lib/t450

/usr/lib/t4014

/usr/lib/vplot

/usr/tmp/raster\$\$

SEE ALSO

plot(3X) in the *Programmer's Reference Manual*.

plot(4), *term(5)* in the *System Administrator's Reference Manual*.



NAME

tput – initialize a terminal or query terminfo database

SYNOPSIS

tput [-Ttype] capname [parms ...]

tput [-Ttype] **init**

tput [-Ttype] **reset**

tput [-Ttype] **longname**

tput -S << *file*

DESCRIPTION

tput uses the *terminfo*(4) database to make the values of terminal-dependent capabilities and information available to the shell (see *sh*(1)), to initialize or reset the terminal, or return the long name of the requested terminal type. *tput* outputs a string if the attribute (*capability name*) is of type string, or an integer if the attribute is of type integer. If the attribute is of type boolean, *tput* simply sets the exit code (0 for TRUE if the terminal has the capability, 1 for FALSE if it does not), and produces no output. Before using a value returned on standard output, the user should test the exit code (\$?, see *sh*(1)) to be sure it is 0. (See EXIT CODES and DIAGNOSTICS below.) For a complete list of capabilities and the *capname* associated with each, see *terminfo*(4).

-Ttype indicates the *type* of terminal. Normally this option is unnecessary, because the default is taken from the environment variable TERM. If **-T** is specified, then the shell variables LINES and COLUMNS and the layer size (see *layers*(1)) will not be referenced.

capname indicates the attribute from the *terminfo*(4) database.

parms If the attribute is a string that takes parameters, the arguments *parms* will be instantiated into the string. An all numeric argument will be passed to the attribute as a number.

-S allows more than one capability per invocation of **tput**. The capabilities must be passed to **tput** from the standard input instead of from the command line (see example). Only one *capname* is allowed per line. The **-S** option changes the meaning of the 0 and 1 boolean and string exit codes (see EXIT CODES).

init If the *terminfo*(4) database is present and an entry for the user's terminal exists (see **-Ttype**, above), the following will occur: (1) if present, the terminal's initialization strings will be output (*is1*, *is2*, *is3*, *if*, *iprogram*), (2) any delays (e.g., newline) specified in the entry will be set in the tty driver, (3) tabs expansion will be turned on or off according to the specification in the entry, and (4) if tabs are not expanded, standard tabs will be set (every 8 spaces). If an entry does not contain the information needed for any of the four above activities, that activity will silently be skipped.

reset Instead of putting out initialization strings, the terminal's reset strings will be output if present (*rs1*, *rs2*, *rs3*, *rf*). If the reset

strings are not present, but initialization strings are, the initialization strings will be output. Otherwise, **reset** acts identically to **init**.

longname If the *terminfo*(4) database is present and an entry for the user's terminal exists (see *-Ttype* above), then the long name of the terminal will be put out. The long name is the last name in the first line of the terminal's description in the *terminfo*(4) database (see *term*(5)).

EXAMPLES

tput init Initialize the terminal according to the type of terminal in the environmental variable **TERM**. This command should be included in everyone's *.profile* after the environmental variable **TERM** has been exported, as illustrated on the *profile*(4) manual page.

tput -T5620 reset Reset an AT&T 5620 terminal, overriding the type of terminal in the environmental variable **TERM**.

tput cup 0 0 Send the sequence to move the cursor to row 0, column 0 (the upper left corner of the screen, usually known as the "home" cursor position).

tput clear Echo the clear-screen sequence for the current terminal.

tput cols Print the number of columns for the current terminal.

tput -T450 cols Print the number of columns for the 450 terminal.

bold= `tput smso `

offbold= `tput rmso `

Set the shell variables **bold**, to begin stand-out mode sequence, and **offbold**, to end standout mode sequence, for the current terminal. This might be followed by a prompt: **echo "\${bold}Please type in your name: \${offbold}\c"**

tput hc Set exit code to indicate if the current terminal is a hard-copy terminal.

tput cup 23 4 Send the sequence to move the cursor to row 23, column 4.

tput longname Print the long name from the *terminfo*(4) database for the type of terminal specified in the environmental variable **TERM**.

tput -S <<! This example shows **tput** processing several capabilities in one invocation. This example clears the screen, moves the cursor to position 10, 10 and turns on bold (extra bright) mode. The list is terminated by an exclamation mark (!) on a line by itself.

FILES

<i>/usr/lib/terminfo/?/*</i>	compiled terminal description database
<i>/usr/include/curses.h</i>	<i>curses</i> (3X) header file
<i>/usr/include/term.h</i>	<i>terminfo</i> (4) header file

/usr/lib/tabset/*

tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs); for more information, see the "Tabs and Initialization" section of *terminfo(4)*

SEE ALSO

stty(1), *tabs(1)*.

profile(4), *terminfo(4)* in the *System Administrator's Reference Manual*.
Chapter 10 of the *Programmer's Guide*.

EXIT CODES

If *capname* is of type boolean, a value of 0 is set for TRUE and 1 for FALSE unless the **-S** option is used.

If *capname* is of type string, a value of 0 is set if the *capname* is defined for this terminal *type* (the value of *capname* is returned on standard output); a value of 1 is set if *capname* is not defined for this terminal *type* (a null value is returned on standard output).

If *capname* is of type boolean or string and the **-S** option is used, a value of 0 is returned to indicate that all lines were successful. No indication of which line failed can be given so exit code 1 will never appear. Exit codes 2, 3, and 4 retain their usual interpretation.

If *capname* is of type integer, a value of 0 is always set, whether or not *capname* is defined for this terminal *type*. To determine if *capname* is defined for this terminal *type*, the user must test the value of standard output. A value of -1 means that *capname* is not defined for this terminal *type*.

Any other exit code indicates an error; see **DIAGNOSTICS**, below.

DIAGNOSTICS

tput prints the following error messages and sets the corresponding exit codes.

exit code	error message
0	-1 (<i>capname</i> is a numeric variable that is not specified in the <i>terminfo(4)</i> database for this terminal <i>type</i> , e.g. tput -T450 lines and tput -T2621 xmc)
1	no error message is printed, see EXIT CODES , above.
2	usage error
3	unknown terminal <i>type</i> or no <i>terminfo(4)</i> database
4	unknown <i>terminfo(4)</i> capability <i>capname</i>



NAME

tr – translate characters

SYNOPSIS

```
tr [ -cds ] [ string1 [ string2 ] ]
```

DESCRIPTION

tr copies the standard input to the standard output with substitution or deletion of selected characters. Input characters found in *string1* are mapped into the corresponding characters of *string2*. Any combination of the options *-cds* may be used:

- c Complements the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 001 through 377 octal.
- d Deletes all input characters in *string1*.
- s Squeezes all strings of repeated output characters that are in *string2* to single characters.

The following abbreviation conventions may be used to introduce ranges of characters or repeated characters into the strings:

- [a–z] Stands for the string of characters whose ASCII codes run from character *a* to character *z*, inclusive.
- [a*n] Stands for *n* repetitions of *a*. If the first digit of *n* is 0, *n* is considered octal; otherwise, *n* is taken to be decimal. A zero or missing *n* is taken to be huge; this facility is useful for padding *string2*.

The escape character \ may be used as in the shell to remove special meaning from any character in a string. In addition, \ followed by 1, 2, or 3 octal digits stands for the character whose ASCII code is given by those digits.

EXAMPLE

The following example creates a list of all the words in *file1* one per line in *file2*, where a word is taken to be a maximal string of alphabets. The strings are quoted to protect the special characters from interpretation by the shell; 012 is the ASCII code for newline.

```
tr -cs "[A-Z][a-z]" "[\012*]" <file1 >file2
```

SEE ALSO

ed(1), sh(1).
ascii(5) in the *System Administrator's Reference Manual*.

BUGS

Will not handle ASCII NUL in *string1* or *string2*; always deletes NUL from input.



NAME

true, *false* – provide truth values

SYNOPSIS

true

false

DESCRIPTION

true does nothing, successfully. *False* does nothing, unsuccessfully. They are typically used in input to *sh(1)* such as:

```
while true
do
    command
done
```

SEE ALSO

sh(1).

DIAGNOSTICS

true has exit status zero, *false* nonzero.



NAME

`tty` - get the name of the terminal

SYNOPSIS

`tty [-1] [-s]`

DESCRIPTION

`tty` prints the path name of the user's terminal.

- `-1` prints the synchronous line number to which the user's terminal is connected, if it is on an active synchronous line.
- `-s` inhibits printing of the terminal path name, allowing one to test just the exit code.

EXIT CODES

- 2 if invalid options were specified,
- 0 if standard input is a terminal,
- 1 otherwise.

DIAGNOSTICS

"not on an active synchronous line" if the standard input is not a synchronous terminal and `-1` is specified.

"not a tty" if the standard input is not a terminal and `-s` is not specified.



NAME

`umask` – set file-creation mode mask

SYNOPSIS

`umask` [*ooo*]

DESCRIPTION

The user file-creation mode mask is set to *ooo*. The three octal digits refer to read/write/execute permissions for *owner*, *group*, and *others*, respectively (see *chmod(2)* and *umask(2)*). The value of each specified digit is subtracted from the corresponding “digit” specified by the system for the creation of a file (see *creat(2)*). For example, `umask 022` removes *group* and *others* write permission (files normally created with mode `777` become mode `755`; files created with mode `666` become mode `644`).

If *ooo* is omitted, the current value of the mask is printed.

`umask` is recognized and executed by the shell.

`umask` can be included in the user’s `.profile` (see *profile(4)*) and invoked at login to automatically set the user’s permissions on files or directories created.

SEE ALSO

`chmod(1)`, `sh(1)`.

`chmod(2)`, `creat(2)`, `umask(2)` in the *Programmer’s Reference Manual*.

`profile(4)` in the *System Administrator’s Reference Manual*.



NAME

uname – print name of current UNIX system

SYNOPSIS

uname [-snrvma]
uname [-S system name]

DESCRIPTION

uname prints the current system name of the UNIX system on the standard output file. It is mainly useful to determine which system one is using. The options cause selected information returned by *uname(2)* to be printed:

- s print the system name (default).
- n print the nodename (the nodename is the name by which the system is known to a communications network).
- r print the operating system release.
- v print the operating system version.
- m print the machine hardware name.
- a print all the above information.

On the 3B2 computer, the system name and the nodename may be changed by specifying a system name argument to the -S option. The system name argument is restricted to 8 characters. Only the super-user is allowed this capability.

SEE ALSO

uname(2) in the *Programmer's Reference Manual*.



NAME

uniq – report repeated lines in a file

SYNOPSIS

uniq [**-udc** [**+n**] [**-n**]] [input [output]]

DESCRIPTION

uniq reads the input file comparing adjacent lines. In the normal case, the second and succeeding copies of repeated lines are removed; the remainder is written on the output file. *Input* and *output* should always be different. Note that repeated lines must be adjacent in order to be found; see *sort*(1). If the **-u** flag is used, just the lines that are not repeated in the original file are output. The **-d** option specifies that one copy of just the repeated lines is to be written. The normal mode output is the union of the **-u** and **-d** mode outputs.

The **-c** option supersedes **-u** and **-d** and generates an output report in default style but with each line preceded by a count of the number of times it occurred.

The *n* arguments specify skipping an initial portion of each line in the comparison:

- n** The first *n* fields together with any blanks before each are ignored. A field is defined as a string of non-space, non-tab characters separated by tabs and spaces from its neighbors.
- +n** The first *n* characters are ignored. Fields are skipped before characters.

SEE ALSO

comm(1), *sort*(1).



NAME

units – conversion program

SYNOPSIS

units

DESCRIPTION

units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

```
You have: inch
You want: cm
          * 2.540000e+00
          / 3.937008e-01
```

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

```
You have: 15 lbs force/in2
You want: atm
          * 1.020689e+00
          / 9.797299e-01
```

units only does multiplicative scale changes; thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

```
pi      ratio of circumference to diameter,
c       speed of light,
e       charge on an electron,
g       acceleration of gravity,
force   same as g,
mole    Avogadro's number,
water   pressure head per unit height of water,
au      astronomical unit.
```

Pound is not recognized as a unit of mass; **lb** is. Compound names are run together, (e.g., **lightyear**). British units that differ from their U.S. counterparts are prefixed thus: **brgallon**. For a complete list of units, type:

```
cat /usr/lib/unittab
```

FILES

```
/usr/lib/unittab
```



NAME

uucp, **uulog**, **uuname** – UNIX-to-UNIX system copy

SYNOPSIS

```
uucp [ options ] source-files destination-file
uulog [ options ] -ssystem
uulog [ options ] system
uulog [ options ] -fssystem
uuname [ -l ] [ -c ]
```

DESCRIPTION

uucp

uucp copies files named by the *source-file* arguments to the *destination-file* argument. A file name may be a path name on your machine, or may have the form:

```
system-name!path-name
```

where *system-name* is taken from a list of system names that *uucp* knows about. The *system-name* may also be a list of names such as

```
system-name!system-name!...!system-name!path-name
```

in which case an attempt is made to send the file via the specified route, to the destination. See WARNINGS and BUGS below for restrictions. Care should be taken to ensure that intermediate nodes in the route are willing to forward information (see WARNINGS below for restrictions).

The following shell metacharacters are disallowed in *system-name*:

```
` ; & | ^ < > ( ) <CR> <TAB> <SPACE>
```

Path names may be one of:

- (1) a full path name;
- (2) a path name preceded by *~user* where *user* is a login name on the specified system and is replaced by that user's login directory;
- (3) a path name preceded by *~/destination* where *destination* is appended to */usr/spool/uucppublic*; (NOTE: This destination will be treated as a file name unless more than one file is being transferred by this request or the destination is already a directory. To ensure that it is a directory, follow the destination with a *'/'*. For example *~/dan/* as the destination will make the directory */usr/spool/uucppublic/dan* if it does not exist and put the requested file(s) in that directory).
- (4) anything else is prefixed by the current directory.

If the result is an erroneous path name for the remote system the copy will fail. If the *destination-file* is a directory, the last part of the *source-file* name is used.

uucp preserves execute permissions across the transmission and gives 0666 read and write permissions (see *chmod(2)*).

The following options are interpreted by *uucp*:

- c** Do not copy local file to the spool directory for transfer to the remote machine (default).
- C** Force the copy of local files to the spool directory for transfer.
- d** Make all necessary directories for the file copy (default).
- f** Do not make intermediate directories for the file copy.
- ggrade** *Grade* is a single letter/number; lower ascii sequence characters will cause the job to be transmitted earlier during a particular conversation.
- j** Output the job identification ASCII string on the standard output. This job identification can be used by *uustat* to obtain the status or terminate a job.
- m** Send mail to the requester when the copy is completed.
- nuser** Notify *user* on the remote system that a file was sent.
- r** Do not start the file transfer, just queue the job.
- sfile** Report status of the transfer to *file*. Note that the *file* must be a full path name.
- xdebug_level** Produce debugging output on standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information. (Debugging will not be available if **uucp** was compiled with **-DSMALL**.)

uulog

uulog queries a log file of *uucp* or *uuxqt* transactions in a file */usr/spool/uucp/.Log/uucico/system*, or */usr/spool/uucp/.Log/uuxqt/system*.

The options cause *uulog* to print logging information:

- ssys** Print information about file transfer work involving system *sys*.
- fsystem** Does a "tail -f" of the file transfer log for *system*. (You must hit BREAK to exit this function.) Other options used in conjunction with the above:
- x** Look in the *uuxqt* log file for the given system.
- number** Indicates that a "tail" command of *number* lines should be executed.

uuname

uuname lists the names of systems known to *uucp*. The **-c** option returns the names of systems known to *cu*. (The two lists are the same, unless your machine is using different *Systems* files for *cu* and *uucp*. See the *Sysfiles* file.) The **-l** option returns the local system name.

FILES

<i>/usr/spool/uucp</i>	spool directories
<i>/usr/spool/uucppublic/*</i>	public directory for receiving and sending (<i>/usr/spool/uucppublic</i>)
<i>/usr/lib/uucp/*</i>	other data and program files

SEE ALSO

mail(1), uustat(1C), uux(1C).
uuxqt(1M) in the *System Administrator's Reference Manual*.
chmod(2) in the *Programmer's Reference Manual*.

WARNINGS

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. You will very likely not be able to fetch files by path name; ask a responsible person on the remote system to send them to you. For the same reasons you will probably not be able to send files to arbitrary path names. As distributed, the remotely accessible files are those whose names begin `/usr/spool/uucppublic` (equivalent to `~/`).

All files received by *uucp* will be owned by *uucp*.

The `-m` option will only work sending files or receiving a single file. Receiving multiple files specified by special shell characters `? * [...]` will not activate the `-m` option.

The forwarding of files through other systems may not be compatible with the previous version of *uucp*. If forwarding is used, all systems in the route must have the same version of *uucp*.

BUGS

Protected files and files that are in protected directories that are owned by the requester can be sent by *uucp* using the `-C` option. However, if the requestor is root, and the directory is not searchable by "other" or the file is not readable by "other", the request will fail.



NAME

`uustat` - `uucp` status inquiry and job control

SYNOPSIS

```

uustat [-a]
uustat [-m]
uustat [-p]
uustat [-q]
uustat [ -kjobid ]
uustat [ -rjobid ]
uustat [ -ssystem ] [ -uuser ]

```

DESCRIPTION

`uustat` will display the status of, or cancel, previously specified `uucp` commands, or provide general status on `uucp` connections to other systems. Only one of the following options can be specified with `uustat` per command execution:

- a Output all jobs in queue.
- m Report the status of accessibility of all machines.
- p Execute a "ps -flp" for all the process-ids that are in the lock files.
- q List the jobs queued for each machine. If a status file exists for the machine, its date, time and status information are reported. In addition, if a number appears in () next to the number of C or X files, it is the age in days of the oldest C./X. file for that system. The Retry field represents the number of hours until the next possible call. The Count is the number of failure attempts. NOTE: for systems with a moderate number of outstanding jobs, this could take 30 seconds or more of real-time to execute. As an example of the output produced by the -q option:

```

eagle      3C    04/07-11:07  NO DEVICES AVAILABLE
mh3bs3     2C    07/07-10:42  SUCCESSFUL

```

The above output tells how many command files are waiting for each system. Each command file may have zero or more files to be sent (zero means to call the system and see if work is to be done). The date and time refer to the previous interaction with the system followed by the status of the interaction.

- k*jobid* Kill the `uucp` request whose job identification is *jobid*. The killed `uucp` request must belong to the person issuing the `uustat` command unless one is the super-user.
- r*jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the limit imposed by the daemon.

Either or both of the following options can be specified with *uustat*:

- ssys** Report the status of all *uucp* requests for remote system *sys*.
- uuser** Report the status of all *uucp* requests issued by *user*.

Output for both the **-s** and **-u** options has the following format:

```
eaglen0000 4/07-11:01:03      (POLL)
eagleN1bd7 4/07-11:07        Seagledan522 /usr/dan/A
eagleC1bd8 4/07-11:07        Seagledan59 D.3b2al2ce4924
                               4/07-11:07    Seagledanrmail mike
```

With the above two options, the first field is the *jobid* of the job. This is followed by the date/time. The next field is either an 'S' or 'R' depending on whether the job is to send or request a file. This is followed by the user-id of the user who queued the job. The next field contains the size of the file, or in the case of a remote execution (*rmail* - the command used for remote mail), the name of the command. When the size appears in this field, the file name is also given. This can either be the name given by the user or an internal name (e.g., D.3b2alce4924) that is created for data files associated with remote executions (*rmail* in this example).

When no options are given, *uustat* outputs the status of all *uucp* requests issued by the current user.

FILES

/usr/spool/uucp/* spool directories

SEE ALSO

uucp(1C).

NAME

uuto, *uupick* – public UNIX-to-UNIX system file copy

SYNOPSIS

uuto [options] source-files destination
uupick [-s system]

DESCRIPTION

uuto sends *source-files* to *destination*. *uuto* uses the *uucp*(1C) facility to send files, while it allows the local system to control the file access. A source-file name is a path name on your machine. Destination has the form:
 system!user

where *system* is taken from a list of system names that *uucp* knows about (see *uuname*). *User* is the login name of someone on the specified system.

Two *options* are available:

- p Copy the source file into the spool directory before transmission.
- m Send mail to the sender when the copy is complete.

The files (or sub-trees if directories are specified) are sent to PUBDIR on *system*, where PUBDIR is a public directory defined in the *uucp* source. By default this directory is /usr/spool/uucppublic. Specifically the files are sent to

PUBDIR/receive/user/mysystem/files.

The destined recipient is notified by *mail*(1) of the arrival of files.

Uupick accepts or rejects the files transmitted to the user. Specifically, *uupick* searches PUBDIR for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from system: [file file-name] [dir dirname] ?

Uupick then reads a line from the standard input to determine the disposition of the file:

- <new-line> Go on to next entry.
- d Delete the entry.
- m [dir] Move the entry to named directory *dir*. If *dir* is not specified as a complete path name (in which \$HOME is legitimate), a destination relative to the current directory is assumed. If no destination is given, the default is the current directory.
- a [dir] Same as m except moving all the files sent from *system*.
- p Print the content of the file.
- q Stop.
- EOT (control-d) Same as q.
- !command Escape to the shell to do *command*.
- * Print a command summary.

Uupick invoked with the -ssystem option will only search the PUBDIR for files sent from *system*.

FILES

PUBDIR /usr/spool/uucppublic public directory

SEE ALSO

mail(1), uucp(1C), uustat(1C), uux(1C).
uucleanup(1M) in the *System Administrator's Reference Manual*.

WARNINGS

In order to send files that begin with a dot (e.g., .profile) the files must be qualified with a dot. For example: .profile, .prof*, .profil? are correct; whereas *prof*, ?profile are incorrect.

NAME

`uux` – UNIX-to-UNIX system command execution

SYNOPSIS

`uux [options] command-string`

DESCRIPTION

`uux` will gather zero or more files from various systems, execute a command on a specified system and then send standard output to a file on a specified system.

NOTE: For security reasons, most installations limit the list of commands executable on behalf of an incoming request from `uux`, permitting only the receipt of mail (see *mail(1)*). (Remote execution permissions are defined in `/usr/lib/uucp/Permissions`.)

The *command-string* is made up of one or more arguments that look like a shell command line, except that the command and file names may be prefixed by *system-name!*. A null *system-name* is interpreted as the local system.

File names may be one of

- (1) a full path name;
- (2) a path name preceded by `~xxx` where `xxx` is a login name on the specified system and is replaced by that user's login directory;
- (3) anything else is prefixed by the current directory.

As an example, the command

```
uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > !~/dan/file.diff"
```

will get the *file1* and *file2* files from the "usg" and "pwba" machines, execute a *diff(1)* command and put the results in *file.diff* in the local `PUBDIR/dan/` directory.

Any special shell characters such as `<>|` should be quoted either by quoting the entire *command-string*, or quoting the special characters as individual arguments.

`uux` will attempt to get all files to the execution system. For files that are output files, the file name must be escaped using parentheses. For example, the command

```
uux a!cut -f1 b!/usr/file \(c!/usr/file\)
```

gets `/usr/file` from system "b" and sends it to system "a", performs a *cut* command on that file and sends the result of the *cut* command to system "c".

`uux` will notify you if the requested command on the remote system was disallowed. This notification can be turned off by the `-n` option. The response comes by remote mail from the remote machine.

The following *options* are interpreted by `uux`:

- The standard input to `uux` is made the standard input to the *command-string*.
- `-aname` Use *name* as the user identification replacing the initiator user-id. (Notification will be returned to the user.)

- b Return whatever standard input was provided to the *uux* command if the exit status is non-zero.
- c Do not copy local file to the spool directory for transfer to the remote machine (default).
- C Force the copy of local files to the spool directory for transfer.
- g*grade* *Grade* is a single letter/number; lower ASCII sequence characters will cause the job to be transmitted earlier during a particular conversation.
- j Output the jobid ASCII string on the standard output which is the job identification. This job identification can be used by *uustat* to obtain the status or terminate a job.
- n Do not notify the user if the command fails.
- p Same as -: The standard input to *uux* is made the standard input to the *command-string*.
- r Do not start the file transfer, just queue the job.
- s*file* Report status of the transfer in *file*.
- x*debug_level* Produce debugging output on the standard output. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information.
- z Send success notification to the user.

FILES

/usr/lib/uucp/spool	spool directories
/usr/lib/uucp/Permissions	remote execution permissions
/usr/lib/uucp/*	other data and programs

SEE ALSO

cut(1), mail(1), uucp(1C), uustat(1C).

WARNINGS

Only the first command of a shell pipeline may have a *system-name!*. All other commands are executed on the system of the first command.

The use of the shell metacharacter * will probably not do what you want it to do. The shell tokens << and >> are not implemented.

The execution of commands on remote systems takes place in an execution directory known to the *uucp* system. All files required for the execution will be put into this directory unless they already reside on that machine. Therefore, the simple file name (without path or machine reference) must be unique within the *uux* request. The following command will NOT work:

```
uux "aldiff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

but the command

```
uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

will work. (If *diff* is a permitted command.)

BUGS

Protected files and files that are in protected directories that are owned by the requestor can be sent in commands using *uux*. However, if the requestor is root, and the directory is not searchable by "other", the request will fail.



NAME

vi – screen-oriented (visual) display editor based on *ex*

SYNOPSIS

vi [-t tag] [-r file] [-L] [-wn] [-R] [-x] [-C] [-c command] file ...
view [-t tag] [-r file] [-L] [-wn] [-R] [-x] [-C] [-c command] file ...
vedit [-t tag] [-r file] [-L] [-wn] [-R] [-x] [-C] [-c command] file ...

DESCRIPTION

vi (visual) is a display-oriented text editor based on an underlying line editor *ex(1)*. It is possible to use the command mode of *ex* from within *vi* and vice-versa. The visual commands are described on this manual page; how to set options (like automatically numbering lines and automatically starting a new output line when you type carriage return) and all *ex(1)* line editor commands are described on the *ex(1)* manual page.

When using *vi*, changes you make to the file are reflected in what you see on your terminal screen. The position of the cursor on the screen indicates the position within the file.

Invocation Options

The following invocation options are interpreted by *vi* (previously documented options are discussed in the NOTES section at the end of this manual page):

- t *tag* Edit the file containing the *tag* and position the editor at its definition.
- r *file* Edit *file* after an editor or system crash. (Recovers the version of *file* that was in the buffer when the crash occurred.)
- L List the name of all files saved as the result of an editor or system crash.
- wn Set the default window size to *n*. This is useful when using the editor over a slow speed line.
- R **Readonly** mode; the **readonly** flag is set, preventing accidental overwriting of the file.
- x Encryption option; when used, *vi* simulates the X command of *ex(1)* and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of *crypt(1)*. The X command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the -x option. See *crypt(1)*. Also, see the WARNING section at the end of this manual page.
- C Encryption option; same as the -x option, except that *vi* simulates the C command of *ex(1)*. The C command is like the X command of *ex(1)*, except that all text read in is assumed to have been encrypted.
- c *command* Begin editing by executing the specified editor *command* (usually a search or positioning command).

The *file* argument indicates one or more files to be edited.

The *view* invocation is the same as *vi* except that the **readonly** flag is set.

The *vedit* invocation is intended for beginners. It is the same as *vi* except that the **report** flag is set to 1, the **showmode** and **novice** flags are set, and **magic** is turned off. These defaults make it easier to learn how to use *vi*.

vi Modes

Command	Normal and initial mode. Other modes return to command mode upon completion. ESC (escape) is used to cancel a partial command.
Input	Entered by setting any of the following options: a A i I o O c C s S R . Arbitrary text may then be entered. Input mode is normally terminated with ESC character, or, abnormally, with an interrupt.
Last line	Reading input for : / ? or ! ; terminate by typing a carriage return; an interrupt cancels termination.

COMMAND SUMMARY

In the descriptions, CR stands for carriage return and ESC stands for the escape key.

Sample commands

← ↓ ↑ →	arrow keys move the cursor
h j k l	same as arrow keys
i <i>text</i> ESC	insert <i>text</i>
cw <i>new</i> ESC	change word to <i>new</i>
easESC	pluralize word (end of word; append s; escape from input state)
x	delete a character
dw	delete a word
dd	delete a line
3dd	delete 3 lines
u	undo previous change
ZZ	exit <i>vi</i> , saving changes
:q!CR	quit, discarding changes
/ <i>text</i> CR	search for <i>text</i>
^U ^D	scroll up or down
:cmdCR	any <i>ex</i> or <i>ed</i> command

Counts before vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of these ways.

line/column number	z G
scroll amount	^D ^U
repeat effect	most of the rest

Interrupting, canceling

ESC	end insert or incomplete cmd
DEL	(delete or rubout) interrupts

File manipulation

ZZ	if file modified, write and exit; otherwise, exit
:wCR	write back changes
:w!CR	forced write, if permission originally not valid
:qCR	quit
:q!CR	quit, discard changes
:e nameCR	edit file <i>name</i>
:e!CR	reedit, discard changes
:e + nameCR	edit, starting at end
:e +nCR	edit starting at line <i>n</i>
:e #CR	edit alternate file
:e! #CR	edit alternate file, discard changes
:w nameCR	write file <i>name</i>
:w! nameCR	overwrite file <i>name</i>
:shCR	run shell, then return
!:cmdCR	run <i>cmd</i> , then return
:nCR	edit next file in arglist
:n argsCR	specify new arglist
^G	show current file and line
:ta tagCR	position cursor to <i>tag</i>

In general, any *ex* or *ed* command (such as *substitute* or *global*) may be typed, preceded by a colon and followed by a carriage return.

Positioning within file

^F	forward screen
^B	backward screen
^D	scroll down half screen
^U	scroll up half screen
nG	go to the beginning of the specified line (end default), where <i>n</i> is a line number
/pat	next line matching <i>pat</i>
?pat	previous line matching <i>pat</i>
n	repeat last / or ? command
N	reverse last / or ? command
/pat/+n	<i>n</i> th line after <i>pat</i>
?pat?-n	<i>n</i> th line before <i>pat</i>
 	next section/function
 	previous section/function
(beginning of sentence
)	end of sentence
{	beginning of paragraph
}	end of paragraph
%	find matching () { or }

Adjusting the screen

^L	clear and redraw window
^R	clear and redraw window if ^L is → key
zCR	redraw screen with current line at top of window

z-CR	⌘	redraw screen with current line at bottom of window
z.CR		redraw screen with current line at center of window
/pat/z-CR		move <i>pat</i> line to bottom of window
zn.CR		use <i>n</i> -line window
^E		scroll window down 1 line
^Y		scroll window up 1 line

Marking and returning

``		move cursor to previous context
''		move cursor to first non-white space in line
mx		mark current position with the ASCII lower-case letter <i>x</i>
`x		move cursor to mark <i>x</i>
ˆx		move cursor to first non-white space in line marked by <i>x</i>

Line positioning

H		top line on screen
L		last line on screen
M		middle line on screen
+		next line, at first non-white
-		previous line, at first non-white
CR		return, same as +
↓ or j		next line, same column
↑ or k		previous line, same column

Character positioning

^		first non white-space character
0		beginning of line
\$		end of line
h or →		forward
l or ←		backward
^H		same as ← (backspace)
space		same as → (space bar)
fx		find next <i>x</i>
Fx		find previous <i>x</i>
tx		move to character prior to next <i>x</i>
Tx		move to character following previous <i>x</i>
;		repeat last f F t or T
,		repeat inverse of last f F t or T
n 		move to column <i>n</i>
%		find matching ({) or }

Words, sentences, paragraphs

w		forward a word
b		back a word
e		end of word
)		to next sentence
}		to next paragraph
(back a sentence
{		back a paragraph
W		forward a blank-delimited word

B back a blank-delimited word
E end of a blank-delimited word

Corrections during insert

^H erase last character (backspace)
^W erase last word
erase your erase character, same as **^H** (backspace)
kill your kill character, erase this line of input
**** quotes your erase and kill characters
ESC ends insertion, back to command mode
DEL interrupt, terminates insert mode
^D backtab one character; reset left margin
 of *autoindent*
^^D caret (^) followed by control-d (^D);
 backtab to beginning of line;
 do not reset left margin of *autoindent*
0^D backtab to beginning of line;
 reset left margin of *autoindent*
^V quote non-printable character

Insert and replace

a append after cursor
A append at end of line
i insert before cursor
I insert before first non-blank
o open line below
O open above
rx replace single char with *x*
RtextESC replace characters

Operators

Operators are followed by a cursor motion, and affect all text that would have been moved over. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator, e.g., **dd** to affect whole lines.

d delete
c change
y yank lines to buffer
< left shift
> right shift
! filter through command

Miscellaneous Operations

C change rest of line (**c\$**)
D delete rest of line (**d\$**)
s substitute chars (**cl**)
S substitute lines (**cc**)
J join lines
x delete characters (**dl**)
X delete characters before cursor (**dh**)

Y yank lines (**yy**)

Yank and Put

Put inserts the text most recently deleted or yanked; however, if a buffer is named (using the ASCII lower-case letters a - z), the text in that buffer is put instead.

3yy yank 3 lines
3yl yank 3 characters
p put back text after cursor
P put back text before cursor
"xp put from buffer *x*
"xy yank to buffer *x*
"xd delete into buffer *x*

Undo, Redo, Retrieve

u undo last change
U restore current line
. repeat last change
"dp retrieve *d*'th last delete

AUTHOR

vi and *ex* were developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

FILES

/tmp default directory where temporary work files are placed; it can be changed using the **directory** option (see the *ex(1)* **set** command)
/usr/lib/terminfo/?/* compiled terminal description database
/usr/lib/.COREterm/?/* subset of compiled terminal description database

NOTES

Two options, although they continue to be supported, have been replaced in the documentation by options that follow the Command Syntax Standard (see *intro(1)*). A **-r** option that is not followed with an option-argument has been replaced by **-L** and **+command** has been replaced by **-c command**.

SEE ALSO

ed(1), *edit(1)*, *ex(1)*.
User's Guide.
Editing Guide.
 curses/terminfo chapter of the *Programmer's Guide*.

WARNINGS

The encryption options are provided with the Security Administration Utilities package, which is available only in the United States.

Tampering with entries in **/usr/lib/.COREterm/?/*** or **/usr/lib/terminfo/?/*** (for example, changing or removing an entry) can affect programs such as *vi(1)* that expect the entry to be present and correct. In particular, removing the "dumb" terminal may cause unexpected problems.

BUGS

Software tabs using `^T` work only immediately after the *autoindent*.

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.



NAME

wait – await completion of process

SYNOPSIS

wait [*n*]

DESCRIPTION

Wait for your background process whose process id is *n* and report its termination status. If *n* is omitted, all your shell's currently active background processes are waited for and the return code will be zero.

The shell itself executes *wait*, without creating a new process.

SEE ALSO

sh(1).

CAVEAT

If you get the error message *cannot fork, too many processes*, try using the *wait(1)* command to clean up your background processes. If this doesn't help, the system process table is probably full or you have too many active foreground processes. (There is a limit to the number of process ids associated with your login, and to the number the system can keep track of.)

BUGS

Not all the processes of a 3- or more-stage pipeline are children of the shell, and thus cannot be waited for.

If *n* is not an active process id, all your shell's currently active background processes are waited for and the return code will be zero.



NAME

wall – write to all users

SYNOPSIS

/etc/wall

DESCRIPTION

wall reads its standard input until an end-of-file. It then sends this message to all currently logged-in users preceded by:

Broadcast Message from ...

It is used to warn all users, typically prior to shutting down the system.

The sender must be super-user to override any protections the users may have invoked (see *mesg(1)*).

FILES

*/dev/tty**

SEE ALSO

mesg(1), *write(1)*.

DIAGNOSTICS

“Cannot send to ...” when the open on a user’s tty file fails.



NAME

`wc` – word count

SYNOPSIS

`wc` [`-lwc`] [*names*]

DESCRIPTION

`wc` counts lines, words, and characters in the named files, or in the standard input if no *names* appear. It also keeps a total count for all named files. A word is a maximal string of characters delimited by spaces, tabs, or new-lines.

The options `l`, `w`, and `c` may be used in any combination to specify that a subset of lines, words, and characters are to be reported. The default is `-lwc`.

When *names* are specified on the command line, they will be printed along with the counts.



NAME

who – who is on the system

SYNOPSIS

who [-uTlHqpdbrtas] [-n x] [file]

who am i

who am I

DESCRIPTION

who can list the user's name, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current UNIX system user. It examines the */etc/utmp* file at login time to obtain its information. If *file* is given, that file (which must be in *utmp*[4] format) is examined. Usually, *file* will be */etc/wtmp*, which contains a history of all the logins since the file was last created.

who with the **am i** or **am I** option identifies the invoking user.

The general format for output is:

```
name [state] line time [idle] [pid] [comment] [exit]
```

The *name*, *line*, and *time* information is produced by all options except **-q**; the *state* information is produced only by **-T**; the *idle* and *pid* information is produced only by **-u** and **-l**; and the *comment* and *exit* information is produced only by **-a**. The information produced for **-p**, **-d**, and **-r** is explained during the discussion of each option, below.

With options, *who* can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the *init* process. These options are:

- u** This option lists only those users who are currently logged in. The *name* is the user's login name. The *line* is the name of the line as found in the directory */dev*. The *time* is the time that the user logged in. The *idle* column contains the number of hours and minutes since activity last occurred on that particular line. A dot (.) indicates that the terminal has seen activity in the last minute and is therefore "current". If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked **old**. This field is useful when trying to determine whether a person is working at the terminal or not. The *pid* is the process-ID of the user's shell. The *comment* is the comment field associated with this line as found in */etc/inittab* (see *inittab*[4]). This can contain information about where the terminal is located, the telephone number of the dataset, type of terminal if hard-wired, etc.
- T** This option is the same as the **-s** option, except that the *state* of the terminal line is printed. The *state* describes whether someone else can write to that terminal. A + appears if the terminal is writable by anyone; a - appears if it is not. **root** can write to all lines having a + or a - in the *state* field. If a bad line is encountered, a ? is printed.

- l This option lists only those lines on which the system is waiting for someone to login. The *name* field is LOGIN in such cases. Other fields are the same as for user entries except that the *state* field does not exist.
- H This option will print column headings above the regular output.
- q This is a quick *who*, displaying only the names and the number of users currently logged on. When this option is used, all other options are ignored.
- p This option lists any other process which is currently active and has been previously spawned by *init*. The *name* field is the name of the program executed by *init* as found in */etc/inittab*. The *state*, *line*, and *idle* fields have no meaning. The *comment* field shows the *id* field of the line from */etc/inittab* that spawned this process. See *inittab*(4).
- d This option displays all processes that have expired and not been respawned by *init*. The *exit* field appears for dead processes and contains the termination and exit values (as returned by *wait*[2]), of the dead process. This can be useful in determining why a process terminated.
- b This option indicates the time and date of the last reboot.
- r This option indicates the current *run-level* of the *init* process. In addition, it produces the process termination status, process id, and process exit status (see *utmp*(4)) under the *idle*, *pid*, and *comment* headings, respectively.
- t This option indicates the last change to the system clock (via the *date*[1] command) by *root*. See *su*(1).
- a This option processes */etc/utmp* or the named *file* with all options turned on.
- s This option is the default and lists only the *name*, *line*, and *time* fields.
- n *x* This option takes a numeric argument, *x*, which specifies the number of users to display per line. *x* must be at least 1. The *-n* option must be used with *-q*.

Note to the super-user: after a shutdown to the single-user state, *who* returns a prompt; the reason is that since */etc/utmp* is updated at login time and there is no login in single-user state, *who* cannot report accurately on this state. *who am i*, however, returns the correct information.

FILES

/etc/utmp
/etc/wtmp
/etc/inittab

SEE ALSO

date(1), *login*(1), *mesg*(1), *su*(1M),
init(1M), *inittab*(4), *utmp*(4) in the *System Administrator's Reference Manual*.
wait(2) in the *Programmer's Reference Manual*.

NAME

write – write to another user

SYNOPSIS

write user [line]

DESCRIPTION

write copies lines from your terminal to that of another user. When first called, it sends the message:

Message from *yourname* (tty??) [date]...

to the person you want to talk to. When it has successfully completed the connection, it also sends two bells to your own terminal to indicate that what you are typing is being sent.

The recipient of the message should write back at this point. Communication continues until an end of file is read from the terminal, an interrupt is sent, or the recipient has executed "mesg n". At that point *write* writes EOT on the other terminal and exits.

If you want to write to a user who is logged in more than once, the *line* argument may be used to indicate which line or terminal to send to (e.g., tty00); otherwise, the first writable instance of the user found in /etc/utmp is assumed and the following message posted:

user is logged on more than one place.

You are connected to "*terminal*".

Other locations are:

terminal

Permission to write may be denied or granted by use of the *mesg(1)* command. Writing to others is normally allowed by default. Certain commands, such as *pr(1)* disallow messages in order to prevent interference with their output. However, if the user has super-user permissions, messages can be forced onto a write-inhibited terminal.

If the character ! is found at the beginning of a line, *write* calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using *write*: when you first *write* to another user, wait for them to *write* back before starting to send. Each person should end a message with a distinctive signal (i.e., (o) for "over") so that the other person knows when to reply. The signal (oo) (for "over and out") is suggested when conversation is to be terminated.

FILES

/etc/utmp	to find user
/bin/sh	to execute !

SEE ALSO

mail(1), mesg(1), pr(1), sh(1), who(1).

DIAGNOSTICS

"*user is not logged on*" if the person you are trying to *write* to is not logged on.

"*Permission denied*" if the person you are trying to *write* to denies that permission (with *mesg*).

WRITE(1)

(Essential Utilities)

WRITE(1)

"Warning: cannot respond, set mesg -y" if your terminal is set to *mesg n* and the recipient cannot respond to you.

"Can no longer write to user" if the recipient has denied permission (*mesg n*) after you had started writing.

NAME

`xargs` – construct argument list(s) and execute command

SYNOPSIS

`xargs` [*flags*] [*command* [*initial-arguments*]]

DESCRIPTION

xargs combines the fixed *initial-arguments* with arguments read from standard input to execute the specified *command* one or more times. The number of arguments read for each *command* invocation and the manner in which they are combined are determined by the flags specified.

command, which may be a shell file, is searched for, using one's \$PATH. If *command* is omitted, `/bin/echo` is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more blanks, tabs, or new-lines; empty lines are always discarded. Blanks and tabs may be embedded as part of an argument if escaped or quoted. Characters enclosed in quotes (single or double) are taken literally, and the delimiting quotes are removed. Outside of quoted strings a backslash (\) will escape the next character.

Each argument list is constructed starting with the *initial-arguments*, followed by some number of arguments read from standard input (Exception: see `-i` flag). Flags `-i`, `-l`, and `-n` determine how arguments are selected for each command invocation. When none of these flags are coded, the *initial-arguments* are followed by arguments read continuously from standard input until an internal buffer is full, and then *command* is executed with the accumulated args. This process is repeated until there are no more args. When there are flag conflicts (e.g., `-l` vs. `-n`), the last flag has precedence. Flag values are:

`-lnumber` *command* is executed for each non-empty *number* lines of arguments from standard input. The last invocation of *command* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first new-line *unless* the last character of the line is a blank or a tab; a trailing blank/tab signals continuation through the next non-empty line. If *number* is omitted, 1 is assumed. Option `-x` is forced.

`-ireplstr` Insert mode: *command* is executed for each line from standard input, taking the entire line as a single arg, inserting it in *initial-arguments* for each occurrence of *replstr*. A maximum of 5 arguments in *initial-arguments* may each contain one or more instances of *replstr*. Blanks and tabs at the beginning of each line are thrown away. Constructed arguments may not grow larger than 255 characters, and option `-x` is also forced. {} is assumed for *replstr* if not specified.

- nnumber** Execute *command* using as many standard input arguments as possible, up to *number* arguments maximum. Fewer arguments will be used if their total size is greater than *size* characters, and for the last invocation if there are fewer than *number* arguments remaining. If option **-x** is also coded, each *number* arguments must fit in the *size* limitation, else *xargs* terminates execution.
- t** Trace mode: The *command* and each constructed argument list are echoed to file descriptor 2 just prior to their execution.
- p** Prompt mode: The user is asked whether to execute *command* each invocation. Trace mode (**-t**) is turned on to print the command instance to be executed, followed by a *?... prompt*. A reply of *y* (optionally followed by anything) will execute the command; anything else, including just a carriage return, skips that particular invocation of *command*.
- x** Causes *xargs* to terminate if any argument list would be greater than *size* characters; **-x** is forced by the options **-i** and **-l**. When neither of the options **-i**, **-l**, or **-n** are coded, the total length of all arguments must be within the *size* limit.
- size** The maximum total size of each argument list is set to *size* characters; *size* must be a positive integer less than or equal to 470. If **-s** is not coded, 470 is taken as the default. Note that the character count for *size* includes one extra character for each argument and the count of characters in the command name.
- eofstr** *eofstr* is taken as the logical end-of-file string. Underbar (*_*) is assumed for the logical EOF string if **-e** is not coded. The value **-e** with no *eofstr* coded turns off the logical EOF string capability (underbar is taken literally). *xargs* reads standard input until either end-of-file or the logical EOF string is encountered.

xargs will terminate if either it receives a return code of **-1** from, or if it cannot execute, *command*. When *command* is a shell program, it should explicitly *exit* (see *sh(1)*) with an appropriate value to avoid accidentally returning with **-1**.

EXAMPLES

The following will move all files from directory \$1 to directory \$2, and echo each move command just before doing it:

```
ls $1 | xargs -i -t mv $1/{} $2/{} 
```

The following will combine the output of the parenthesized commands onto one line, which is then echoed to the end of file *log*:

```
(logname; date; echo $0 $*) | xargs >>log
```

The user is asked which files in the current directory are to be archived and archives them into *arch* (1.) one at a time, or (2.) many at a time.

1. `ls | xargs -p -l ar r arch`
2. `ls | xargs -p -l | xargs ar r arch`

The following will execute *diff*(1) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs -n2 diff
```

SEE ALSO

`sh`(1).



Index to Utilities

■ 2K File System Utilities

ff	ff(1M)
finc	finc(1M)
frec	frec(1M)
fsba	fsba(1M)
mkfs	mkfs(1M)

■ ASSIST Utilities

assist	assist(1)
astgen	astgen(1)

■ AT&T Windowing Utilities

ismpx	ismpx(1)
jterm	jterm(1)
jwin	jwin(1)
layers	layers(1)
relogin	relogin(1M)
xtd	xtd(1M)
xts	xts(1M)
xtt	xtt(1M)

■ Basic Networking Utilities

ct	ct(1C)
cu	cu(1C)
uucico	uucico(1M)
uucp	uucp(1C)
uugetty	uugetty(1C)
uulog	uucp(1C)
uuname	uucp(1C)
uupick	uuto(1C)
uustat	uustat(1C)
uuto	uuto(1C)
uux	uux(1C)

■ Cartridge Tape Utilities

cmpress	cmpress(1M)
ctccpio	ctccpio(1M)
ctcfmt	ctcfmt(1M)
ctcinfo	ctcinfo(1M)
finc	finc(1M)
frec	frec(1M)

tar tar(1)

■ **Directory and File Management Utilities**

ar ar(1)
awk awk(1)
bdiff bdiff(1)
bfs bfs(1)
col col(1)
comm comm(1)
csplit csplit(1)
cut cut(1)
diff3 diff3(1)
dircmp dircmp(1)
egrep egrep(1)
fgrep fgrep(1)
find find(1)
join join(1)
nawk nawk(1)
newform newform(1)
nl nl(1)
od od(1)
pack pack(1)
paste paste(1)
pcat pack(1)
pg pg(1)
sdiff sdiff(1)
split split(1)
sum sum(1)
tail tail(1)
touch touch(1)
tr tr(1)
uniq uniq(1)
unpack pack(1)

■ **Editing Utilities**

edit edit(1)
ex ex(1)
vedit vi(1)
vi vi(1)
view vi(1)

■ Essential Utilities

brc	brc(1M)
cat	cat(1)
cd	cd(1)
checkfsys	checkfsys(1M)
chgrp	chown(1)
chmod	chmod(1)
chown	chown(1)
chroot	chroot(1M)
chrtbl	chrtbl(1M)
clri	clri(1M)
cmp	cmp(1)
cp	cp(1)
cpio	cpio(1)
crash	crash(1M)
cron	cron(1M)
date	date(1)
dcopy	dcopy(1M)
dd	dd(1M)
devnm	devnm(1M)
df	df(1M)
diff	diff(1)
drvinstall	drvinstall(1M)
dsconfig	dsconfig(1M)
du	du(1M)
echo	echo(1)
ed	ed(1)
editsa	editsa(1M)
edittbl	edittbl(1M)
errdump	errdump(1M)
expr	expr(1)
false	true(1)
ff	ff(1M)
file	file(1)
fmtflop	fmtflop(1M)
fmthard	fmthard(1M)
format	format(1M)
fsck	fsck(1M)
fsdb	fsdb(1M)
fsstat	fsstat(1M)
fstyp	fstyp(1M)

fuser	fuser(1M)
getmajor	getmajor(1M)
getopt	getopt(1)
getoptcv	getopts(1)
getopts	getopts(1)
getty	getty(1M)
grep	grep(1)
hdeadd	hdeadd(1M)
hdefix	hdefix(1M)
hdelogger	hdelogger(1M)
id	id(1)
init	init(1M)
kill	kill(1)
killall	killall(1M)
labelit	labelit(1M)
ldsysdump	ldsysdump(1M)
led	led(1M)
line	line(1)
link	link(1M)
ln	cp(1)
login	login(1)
ls	ls(1)
magic	magic(1M)
mail	mail(1)
mailx	mailx(1)
makefsys	makefsys(1M)
mesg	mesg(1)
mkboot	mkboot(1M)
mkdir	mkdir(1)
mkfs	mkfs(1M)
mknod	mknod(1M)
mkunix	mkunix(1M)
mount	mount(1M)
mountall	mountall(1M)
mountfsys	mountfsys(1M)
mv	cp(1)
mvdir	mvdir(1M)
ncheck	ncheck(1M)
newboot	newboot(1M)
newgrp	newgrp(1M)
news	news(1)

passmgmt passmgmt(1M)
passwd passwd(1)
powerdown powerdown(1M)
pr pr(1)
prtconf prtconf(1M)
prvtoc prvtoc(1M)
ps ps(1)
pump pump(1M)
pwck pwck(1M)
pwconv pwconv(1M)
pwd pwd(1)
pwunconv pwunconv(1M)
rc0 rc0(1M)
rc2 rc2(1M)
rc3 rc3(1M)
red ed(1)
rm rm(1)
rmail mail(1)
rmdir rm(1)
rsh sh(1)
sed sed(1)
setclk setclk(1M)
setmnt setmnt(1M)
setup setup(1)
sh sh(1)
shutdown shutdown(1M)
sleep sleep(1)
sort sort(1)
stty stty(1)
su su(1M)
swap swap(1M)
sync sync(1M)
sysadm sysadm(1)
sysdef sysdef(1M)
tail tail(1)
tee tee(1)
telinit telinit(1M)
test test(1)
touch touch(1M)
true true(1)
u3b2 machid(1)

uadmin	uadmin(1M)
umask	umask(1)
umount	umount(1M)
umountall	umountall(1M)
uname	uname(1)
volcopy	volcopy(1M)
wait	wait(1)
wall	wall(1)
wc	wc(1)
what	what(1M)
who	who(1)
whodo	whodo(1M)
write	write(1)

■ **Form and Menu Language Interpreter Utilities**

coproc	coproc(1F)
echo	echo(1F)
getfrm	getfrm(1F)
getitems	getitems(1F)
indicator	indicator(1F)
indicator	indicator(1F)
message	message(1F)
pathconv	pathconv(1F)
readfile	readfile(1F)
regex	regex(1F)
reinit	reinit(1F)
reset	reset(1F)
run	run(1F)
set	set(1F)
setcolor	setcolor(1F)
shell	shell(1F)
vsig	vsig(1F)

■ **Inter-process Communications Utilities**

ipcrm	ipcrm(1)
ipcs	ipcs(1)

■ **Line Printer Spooling Utilities**

cancel	lp(1)
disable	enable(1)
enable	enable(1)
lp	lp(1)
lpadmin	lpadmin(1M)

lpfilter lpfilter(1M)
 lpforms lpforms(1M)
 lpsched lpsched(1M)
 lpstat lpstat(1)
 lpusers lpusers(1M)

■ Networking Support Utilities

nlsadmin nlsadmin(1M)
 strace strace(1M)
 strclean strclean(1M)
 strerr strerr(1M)

■ Remote File Sharing Utilities

adv adv(1M)
 dname dname(1M)
 fumount fumount(1M)
 fusage fusage(1M)
 idload idload(1M)
 nsquery nsquery(1M)
 rfadmin rfadmin(1M)
 rfpaswd rfpaswd(1M)
 rfstart rfstart(1M)
 rfstop rfstop(1M)
 rmnttry rmnttry(1M)
 rmount rmount(1M)
 rmountall rmountall(1M)
 rumount rumount(1M)
 unadv unadv(1M)

■ Security Administration Utilities

crypt crypt(1)
 makekey makekey(1)

■ Spell Utilities

deroff deroff(1)
 spell spell(1)
 /usr/lib/hashcheck spell(1)
 /usr/lib/hashmake spell(1)
 /usr/lib/spellin spell(1)

■ System Performance Analysis Utilities

graph graph(1G)
 prf prf(7)
 profiler profiler(1M)

Index to Utilities

sadp sadp(1M)
sag sag(1G)
sar sar(1)
sar sar(1M)
timex timex(1)
tplot tplot(1G)

■ Terminal Information Utilities

captainfo captainfo(1M)
infocmp infocmp(1M)
tic tic(1M)
tput tput(1)

■ User Environment Utilities

at at(1)
banner banner(1)
basename basename(1)
batch at(1)
bc bc(1)
cal cal(1)
calendar calendar(1)
crontab crontab(1)
dc dc(1)
dirname basename(1)
env env(1)
factor factor(1)
line line(1)
logname logname(1)
nice nice(1)
nohup nohup(1)
shl shl(1)
tabs tabs(1)
time time(1)
tty tty(1)
u3b machid(1)
u3b5 machid(1)
units units(1)
vax machid(1)
xargs xargs(1)

Introduction

This *System Administrator's Reference Manual* describes the commands, file formats, and miscellaneous facilities used by those who administer a UNIX system running on the AT&T 3B2 Computer.

Several closely-related documents contain other valuable information:

- The *System Administrator's Guide* provides procedures for and explanations of administrative tasks.
- The *User's Guide* presents an overview of the UNIX system and tutorials on how to use text editors, automate repetitive jobs, and send information to others.
- The *User's Reference Manual* describes the commands that constitute the basic software running on the AT&T 3B2 Computer.
- The *Programmer's Guide* presents an overview of the UNIX system programming environment and tutorials on various programming tools.
- The *Programmer's Reference Manual* describes the commands, system calls, subroutines, libraries, file formats, and miscellaneous facilities used by programmers.

The *System Administrator's Reference Manual* is divided into five sections:

- (1M) System Maintenance Commands and Application Programs
- (4) File Formats
- (5) Miscellaneous Facilities
- (7) Special Files
- (8) System Maintenance Procedures

Throughout this manual, numbers following a command are intended for easy cross-reference. A command with a (1M), (7), or (8) following it means that the command is in the corresponding section of this manual. A command with a (4) or (5) following it means that the command is in the corresponding section of this manual and the *Programmer's Reference Manual*. A command followed by a (1), (1C), or (1G) usually means that it is found in the *User's Reference Manual*. (Section 1 commands appropriate for use by programmers are located in the *Programmer's Reference Manual*.) A command with a (2) or (3) following it means that the command is in the

corresponding section of the *Programmer's Reference Manual*.

Section 1M, "System Maintenance Commands and Application Programs," contains commands and programs that are used in administering a UNIX system.

Section 4, "File Formats," documents the structure of particular kinds of files. For example, the format of `/etc/passwd` is given on *passwd(4)* and the content of `/etc/profile` is explained on *profile(4)*. In general, when a C language structure corresponds to a file format, it can be found in either the `/usr/include` or `/usr/include/sys` directories.

Section 5, "Miscellaneous Facilities," contains a variety of information. For example, a table of the octal and hexadecimal equivalents of the ASCII character set is given on *ascii(5)*, shell environmental variables (such as `HOME`, `PATH`, `LANGUAGE`, etc.) are described on *environ(5)*, and names of common AT&T terminals are listed on *term(5)*.

Section 7, "Special Files," discusses the characteristics of system files that refer to input/output devices. The names in this section generally refer to device names for the hardware, rather than to the names of the special files themselves.

Section 8, "System Maintenance Procedures," discusses crash recovery, firmware programs, boot procedures, facility descriptions, etc.

Each section begins with a page labeled *intro*. Entries following the *intro* page are arranged alphabetically and may consist of more than one page. Some entries describe several routines, commands, etc. In such cases, the entry appears only once, alphabetized under its "primary" name. (An example of such an entry is **mount(1M)**, which also describes the **umount** command.) The "secondary" commands are listed directly below their associated primary command. To learn which manual page describes a secondary command, locate its name in the middle column of the "Permuted Index" and follow across that line to the name of the manual page listed in the right column.

All entries are based on a common format, not all of whose parts always appear:

- **NAME** gives the name(s) of the entry and briefly states its purpose.
- **SYNOPSIS** summarizes the use of the program being described. A few conventions are used, particularly in Section 1M (*Commands*):

- **Boldface** strings are literals and are to be typed just as they appear.
 - *Italic* strings usually represent substitutable argument and program names found elsewhere in the manual. (They are underlined in the typed version of the entries.)
 - Square brackets **[]** around an argument indicate that the argument is optional. When an argument is given as "name" or "file," it always refers to a *file* name.
 - Ellipses ... are used to show that the previous argument may be repeated.
- **DESCRIPTION** provides an overview of the command.
 - **EXAMPLE(S)** gives example(s) of usage, where appropriate.
 - **FILES** gives the file names that are built into the program.
 - **SEE ALSO** offers pointers to related information.
 - **DIAGNOSTICS** discusses the diagnostic indications that may be produced. Messages that are intended to be self-explanatory are not listed.
 - **WARNINGS** points out potential pitfalls.
 - **BUGS** gives known bugs and sometimes deficiencies.

Preceding Section 1 are a "Table of Contents" (listing both primary and secondary command entries) and a "Permuted Index." Each line of the "Table of Contents" contains the name of a manual page (with secondary entries, if they exist) and an abstract of that page. Each line of the "Permuted Index" represents a permutation (or sorting) of a line from the "Table of Contents" into three columns. The lines are arranged so that a keyword or phrase begins the middle column. Use the "Permuted Index" by searching this middle column for a topic or command. When you have found the entry you want, the right column of that line lists the name of the manual page on which information corresponding to that keyword can be found. The left column contains the remainder of the permutation that began in the middle column.



Table of Contents

1. System Maintenance Commands and Application Programs

intro(1M)	introduction to maintenance commands and application programs
accept, reject(1M)	allow or prevent LP requests
adv(1M)	advertise a directory for remote access
brc, bcheckrc(1M)	system initialization procedures
captainfo(1M)	convert a termcap description into a terminfo description
checkfsys(1M)	check a file system on a removable disk
chroot(1M)	change root directory for a command
chrtbl(1M)	generate character classification and conversion tables
ckauto(1M)	find if the UNIX system was reconfigured at boot time
ckbupscd(1M)	check file system backup schedule
clri(1M)	clear i-node
cmpress(1M)	re-link file system to remove fragmentation
crash(1M)	examine system images
cron(1M)	clock daemon
ctccpio(1M)	copy file archives in and out to cartridge tape
ctcfmt(1M)	format cartridge tape
ctcinfo(1M)	display information about cartridge tape
dcopy(1M)	copy file systems for optimal access time
dd(1M)	convert and copy a file
devinfo(1M)	print device specific information
devnm(1M)	device name
df(1M)	report number of free disk blocks and i-nodes
disks(1M)	adds /dev/entries for hard disks in the Equipped Device Table
dname(1M)	Print Remote File Sharing domain and network names
drvinstall(1M)	install/uninstall a driver
du(1M)	summarize disk usage
editsa(1M)	add/delete entry from software application file
edittbl(1M)	edit edt_data file
errdump(1M)	print error log
ff(1M)	list file names and statistics for a file system
finc(1M)	fast incremental backup
fltboot(1M)	set NVRAM parameters for floating boot
fntflop(1M)	physically format diskettes
fmthard(1M)	populate VTOC on hard disks
format(1M)	physically format a SCSI hard disk
frec(1M)	recover files from a backup tape
fsba(1M)	file system block analyzer

Table of Contents

fsck, dfsc(1M)	check and repair file systems
fsdb(1M)	file system debugger
fsstat(1M)	report file system status
fstyp(1M)	determine file system identifier
fumount(1M)	forced unmount of an advertised resource
fusage(1M)	disk access profiler
fuser(1M)	identify processes using a file or file structure
genc(1M)	create a front-end to the cc command
getmajor(1M)	print major number(s) of hardware devices
getty(1M)	set terminal type, modes, speed, and line discipline
hdeadd(1M)	add/delete hdelog (Hard Disk Error Log) reports
hdefix(1)	report or change bad block mapping on a hard disk device
hdelogger(1M)	Hard Disk Error status report command and Log Daemon
id(1M)	print user and group IDs and names
idload(1M)	Remote File Sharing user and group mapping
infocmp(1M)	compare or print out terminfo descriptions
init, telinit(1M)	process control initialization
install(1M)	install commands
killall(1M)	kill all active processes
labelit(1M)	provide labels for file systems
ldsysdump(1M)	load system dump from floppy diskettes
led(1M)	flash green LED
link, unlink(1M)	link and unlink files and directories
lpadmin(1M)	configure the LP print service
lpfilter(1M)	administer filters used with the LP print service
lpforms(1M)	administer forms used with the LP print service
lpsched, lpshut, lpmove(1M)	start/stop the LP print service and move requests
lpusers(1M)	set printing queue priorities
makefsys(1M)	create a file system on a diskette
makehdfsys(1M)	construct file systems on hard disks
mkboot(1M)	convert an object file to a bootable object file
mkfs(1M)	construct a file system
mknod(1M)	build special file
mkunix(1M)	make a bootable system file with kernel and driver symbol tables
mount, umount(1M)	mount and unmount file systems and remote resources
mountall, umountall(1M)	mount, unmount multiple file systems
mountfsys, umountfsys(1M)	mount, unmount a diskette file system
mmdir(1M)	move a directory
ncheck(1M)	generate path names from i-numbers
newboot(1M)	load olboot and mboot onto the disk boot partition
newgrp(1M)	log in to a new group

nlsadmin(1M)	network listener service administration
nsquery(1M)	Remote File Sharing name server query
passgmt(1M)	password files management
powerdown(1M)	stop all processes and turn off the power
profiler: prfld, prfstat, prfdc, prfsnap, prfpr(1M)	UNIX system profiler
prtconf(1M)	print system configuration
prvtoc(1M)	print the VTOC of a block device
pump(1M)	Download B16 or X86 a.out file to a peripheral board
pwck, grpck(1M)	password/group file checkers
pwconv(1M)	Installs and updates
pwunconv(1M)	Converts from a two- to a one-password file scheme
rc0(1M)	run commands performed to stop the operating system
rc2(1M)	run commands performed for multi-user environment
relogin(1M)	rename login entry to show current layer
rfadmin(1M)	Remote File Sharing administration
rfpasswd(1M)	change Remote File Sharing host password
rfstart(1M)	start Remote File Sharing
rfstop(1M)	stop the Remote File Sharing environment
rfuadmin(1M)	Remote File Sharing notification shell script
rfudaemon(1M)	Remote File Sharing daemon process
rmntstat(1M)	display mounted resource information
rmnttry(1M)	attempt to mount queued remote resources
rmount(1M)	queue remote resource mounts
rmountall, rumountall(1M)	mount, unmount Remote File Sharing resources
rumount(1M)	cancel queued remote resource request
sadp(1M)	disk access profiler
sar: sa1, sa2, sadc(1M)	system activity report package
setclk(1M)	set system time from hardware clock
setmnt(1M)	establish mount table
shutdown(1M)	shut down system, change system state
strace(1M)	print STREAMS trace messages
strclean(1M)	STREAMS error logger cleanup program
strerr(1M)	STREAMS error logger daemon
su(1M)	become super-user or another user
swap(1M)	swap administrative interface
sync(1M)	update the super block
sysadm(1)	menu interface to do system administration
sysdef(1M)	output system definition
tic(1M)	terminfo compiler
uadmin(1M)	administrative control
unadv(1M)	unadvertise a Remote File Sharing resource

Table of Contents

uuccheck(1M)	check the uucp directories and permissions file
uucico(1M)	file transport program for the uucp system
uucleanup(1M)	uucp spool directory clean-up
uugetty(1M)	set terminal type, modes, speed, and line discipline
uusched(1M)	the scheduler for the uucp file transport program
Uutry(1M)	try to contact remote system with debugging on
uuxqt(1M)	execute remote command requests
volcopy(1M)	make literal copy of file system
whodo(1M)	who is doing what
wtinit(1M)	object downloader for the 5620 DMD terminal
xtd(1M)	extract and print xt driver link structure
xts(1M)	extract and print xt driver statistics
xtt(1M)	extract and print xt driver packet traces

4. File Formats

intro(4)	introduction to file formats
pathalias(4)	alias file for FACE
a.out(4)	common assembler and link editor output
acct(4)	per-process accounting file format
ar(4)	common archive file format
cftime(4)	language specific strings
checklist(4)	list of file systems processed by fsck and ncheck
core(4)	format of core image file
cpio(4)	format of cpio archive
dir(4)	format of directories
dirent(4)	file system independent directory entry
.edt_swapp(4)	software application file
.environ(4)	system-wide FACE environment variables
filehdr(4)	file header for common object files
fs: file system(4)	format of system volume
fspec(4)	format specification in text files
fstab(4)	file-system-table
gettydefs(4)	speed and terminal settings used by getty
gps(4)	graphical primitive string, format of graphical files
group(4)	group file
inittab(4)	script for the init process
inode(4)	format of an i-node
issue(4)	issue identification file
ldfcn(4)	common object file access routines
limits(4)	file header for implementation-specific constants
linenum(4)	line number entries in a common object file

/usr/adm/loginlog(4) log of failed login attempts
 master(4) master configuration database
 mnttab(4) mounted file system table
 .ott(4) files that hold object architecture information
 passwd(4) password file
 plot(4) graphics interface
 pnch(4) file format for card images
 profile(4) setting up an environment at login time
 reloc(4) relocation information for a common object file
 rfmaster(4) Remote File Sharing name server master file
 sccsfile(4) format of SCCS file
 scnhdr(4) section header for a common object file
 scr_dump(4) format of curses screen image file
 syms(4) common object file symbol table format
 system(4) system configuration information table
 term(4) format of compiled term file
 terminfo(4) terminal capability data base
 timezone(4) set default system time zone
 unistd(4) file header for symbolic constants
 utmp, wtmp(4) utmp and wtmp entry formats

5. Miscellaneous Facilities

intro(5) introduction to miscellany
 ascii(5) map of ASCII character set
 environ(5) user environment
 fcntl(5) file control options
 jagent(5) host control of windowing terminal
 layers(5) protocol used between host and windowing terminal under
 math(5) math functions and constants
 prof(5) profile within a function
 regexp(5) regular expression compile and match routines
 stat(5) data returned by stat system call
 term(5) conventional names for terminals
 types(5) primitive system data types
 values(5) machine-dependent values
 varargs(5) handle variable argument list
 xtproto(5) multiplexed channels protocol used by xt(7) driver

7. Special Files

intro(7) introduction to special files
 clone(7) open any minor device on a STREAMS driver

Table of Contents

console(7)	console interface
hdelog(7)	hard disk error log interface file
id(7)	3B2 computer Integral Disk Subsystem
if(7)	3B2 computer Floppy Disk Subsystem
log(7)	interface to STREAMS error logging and event tracing
mem, kmem(7)	core memory
mt(7)	tape interface
null(7)	the null file
ports(7)	5 line asynchronous interface
prf(7)	operating system profiler
SA(7)	devices administered by System Administration
streamio(7)	STREAMS ioctl commands
sxt(7)	pseudo-device driver
termio(7)	general terminal interface
timod(7)	Transport Interface cooperating STREAMS module
tirdwr(7)	Transport Interface read/write interface STREAMS module
tty(7)	controlling terminal interface
xt(7)	multiplexed tty driver for AT&T windowing terminals

8. System Maintenance Procedures

intro(8)	introduction to system maintenance procedures
3B2boot(8)	3B2 computer bootstrap procedures
baud(8)	display or change console baud rate
dgmon(8)	run diagnostic phases in firmware
edt(8)	Equipped Device Table commands
filledt(8)	fill Equipped Device Table (EDT)
mk(8)	remake the binary system and commands from source code
newkey(8)	makes a floppy key for the 3B2 Computer
passwd(8)	change firmware password
ports(8)	create character device files and inittab entries for ports boards
sysdump(8)	boot option to dump system memory image to floppy disk(s)
version(8)	version commands

Permuted Index

3B2boot(8) 3B2 computer bootstrap procedures 3B2boot(8)
 if(7) 3B2 computer Floppy Disk Subsystem if(7)
 Subsystem id(7) 3B2 computer Integral Disk id(7)
 makes a floppy key for the 3B2 Computer newkey(8) newkey(8)
 procedures 3B2boot(8) 3B2 computer bootstrap 3B2boot(8)
 ports(7) 5 line asynchronous interface ports(7)
 object downloader for the 5620 DMD terminal wtinit(1M) wtinit(1M)
 prevent LP requests accept(1M) reject(1M) allow or accept(1M)
advertise a directory for remote access adv(1M) adv(1M)
 fusage(1M) disk access profiler fusage(1M)
 sadb(1M) disk access profiler sadp(1M)
ldfcn(4) common object file access routines ldfcn(4)
copy file systems for optimal access time dcopy(1M) dcopy(1M)
 acct(4) per-process accounting file format acct(4)
 format acct(4) per-process accounting file acct(4)
 killall(1M) kill all active processes killall(1M)
sa1(1M) sa2(1M) sadc(1M) system activity report package sar(1M) sar(1M)
 application file editsa(1M) add/delete entry from software editsa(1M)
 Log) reports hdeadd(1M) add/delete hdelog (Hard Disk Error hdeadd(1M)
the Equipped Device/ disks(1M) adds /dev/entries for hard disks in disks(1M)
 print service lpfilter(1M) administer filters used with the LP lpfilter(1M)
 print service lpforms(1M) administer forms used with the LP lpforms(1M)
Administration SA(7) devices administered by System SA(7)
 network listener service administration nlsadmin(1M) nlsadmin(1M)
rfadmin(1M) Remote File Sharing administration rfadmin(1M)
 devices administered by System Administration SA(7) SA(7)
 menu interface to do system administration sysadm(1) sysadm(1)
 uadmin(1M) administrative control uadmin(1M)
 swap(1M) swap administrative interface swap(1M)
 remote access adv(1M) advertise a directory for adv(1M)
 access adv(1M) advertise a directory for remote adv(1M)
fumount(1M) forced unmount of an advertised resource fumount(1M)
 pathalias(4) alias file for FACE pathalias(4)
 accept(1M) reject(1M) allow or prevent LP requests accept(1M)
 fsba(1M) file system block analyzer fsba(1M)
pump(1M) Download B16 or X86 a.out file to a peripheral board pump(1M)
 editor output a.out(4) common assembler and link a.out(4)
add/delete entry from software application file editsa(1M) editsa(1M)
 .edt_swapp(4) software application file edt_swapp(4)
to maintenance commands and application programs /introduction intro(1M)
 .ar(4) common archive file format ar(4)
 architecture information &.ott(4)
 archive cpio(4)
 archive file format ar(4)
 archives in and out to cartridge ctccpio(1M)
 argument list varargs(5)
 ASCII character set ascii(5)

Permuted Index

a.out(4) common
 ports(7) 5 line
xt(7) multiplexed tty driver for
 resources rmntry(1M)
peripheral board pump(1M) Download
 finc(1M) fast incremental
ckbupscd(1M) check file system
 frec(1M) recover files from a
device hdefix(1) report or change
 terminal capability data
baud(8) display or change console
 baud rate
 procedures brc(1M)
 su(1M)
 source code mk(8) remake the
 fsba(1M) file system
prtvtoc(1M) print the VTOC of a
 hdefix(1) report or change bad
 sync(1M) update the super
df(1M) report number of free disk
 or X86 a.out file to a peripheral
 files and inittab entries for ports
set NVRAM parameters for floating
image to floppy disk(s) sysdump(8)
load olboot and mboot onto the disk
the UNIX system was reconfigured at
 convert an object file to a
 and driver/ mkunix(1M) make a
 3B2boot(8) 3B2 computer
 initialization procedures
 mknod(1M)
 data returned by stat system
 request rumount(1M)
 terminfo(4) terminal
 description into a terminfo/
 pnch(4) file format for
copy file archives in and out to
 ctcfmt(1M) format
 display information about
gencc(1M) create a front-end to the
 disk device hdefix(1) report or
 baud(8) display or
 passwd(8)
 password rpasswd(1M)
 chroot(1M)
shutdown(1M) shut down system,
 driver xtproto(5) multiplexed
 ascii(5) map of ASCII character set ascii(5)
 assembler and link editor output a.out(4)
 asynchronous interface ports(7)
 AT&T windowing terminals xt(7)
 attempt to mount queued remote rmntry(1M)
 B16 or X86 a.out file to a pump(1M)
 backup finc(1M)
 backup schedule ckbupscd(1M)
 backup tape frec(1M)
 bad block mapping on a hard disk hdefix(1)
 base terminfo(4) terminfo(4)
 baud rate baud(8)
 baud(8) display or change console baud(8)
 bcheckrc(1M) system initialization brc(1M)
 become super-user or another user su(1M)
 binary system and commands from mk(8)
 block analyzer fsba(1M)
 block device prtvtoc(1M)
 block mapping on a hard disk device hdefix(1)
 block sync(1M)
 blocks and i-nodes df(1M)
 board pump(1M) Download B16 pump(1M)
 boards /create character device ports(8)
 boot fltboot(1M) fltboot(1M)
 boot option to dump system memory sysdump(8)
 boot partition newboot(1M) newboot(1M)
 boot time ckauto(1M) find if ckauto(1M)
 bootable object file mkboot(1M) mkboot(1M)
 bootable system file with kernel mkunix(1M)
 bootstrap procedures 3B2boot(8)
 brc(1M) bcheckrc(1M) system brc(1M)
 build special file mknod(1M)
 call stat(5) stat(5)
 cancel queued remote resource rumount(1M)
 capability data base terminfo(4)
 captainfo(1M) convert a termcap captainfo(1M)
 card images pnch(4)
 cartridge tape ctccpio(1M) ctccpio(1M)
 cartridge tape ctcfmt(1M)
 cartridge tape ctinfo(1M) ctinfo(1M)
 cc command gencc(1M)
 cftime(4) language specific strings cftime(4)
 change bad block mapping on a hard hdefix(1)
 change console baud rate baud(8)
 change firmware password passwd(8)
 change Remote File Sharing host rpasswd(1M)
 change root directory for a command chroot(1M)
 change system state shutdown(1M)
 channels protocol used by xt(7) xtproto(5)

conversion/ chrtbl(1M) generate character classification and chrtbl(1M)
 entries for ports/ ports(8) create character device files and inittab ports(8)
 ascii(5) map of ASCII character set ascii(5)
 disk checkfsys(1M) check a file system on a removable checkfsys(1M)
 fsck(1M) dfscck(1M) check and repair file systems fsck(1M)
 ckbupscd(1M) check file system backup schedule ckbupscd(1M)
 permissions file uucheck(1M) check the uucp directories and uucheck(1M)
 grpck(1M) password/group file checkers pwck(1M) pwck(1M)
 on a removable disk checkfsys(1M) check a file system checkfsys(1M)
 processed by fsck and ncheck checklist(4) list of file systems checklist(4)
 for a command chroot(1M) change root directory chroot(1M)
 classification and conversion/ chrtbl(1M) generate character chrtbl(1M)
 was reconfigured at boot time ckauto(1M) find if the UNIX system ckauto(1M)
 backup schedule ckbupscd(1M) check file system ckbupscd(1M)
 chrtbl(1M) generate character classification and conversion/ chrtbl(1M)
 strclean(1M) STREAMS error logger cleanup program strclean(1M)
 uucleanup(1M) uucp spool directory clean-up uucleanup(1M)
 clri(1M) clear i-node clri(1M)
 cron(1M) clock daemon cron(1M)
 set system time from hardware clock setclk(1M) setclk(1M)
 STREAMS driver clone(7) open any minor device on a clone(7)
 remove fragmentation clri(1M) clear i-node clri(1M)
 system and commands from source cmpress(1M) re-link file system to cmpress(1M)
 /Hard Disk Error status report code mk(8) remake the binary mk(8)
 change root directory for a command and Log Daemon hdelogger(1M)
 create a front-end to the cc command chroot(1M) chroot(1M)
 uuxqt(1M) execute remote command gencc(1M) gencc(1M)
 /introduction to maintenance command requests uuxqt(1M)
 edt(8) Equipped Device Table commands and application programs intro(1M)
 mk(8) remake the binary system and commands edt(8)
 install(1M) install commands from source code mk(8)
 environment rc2(1M) run commands install(1M)
 operating system rc0(1M) run commands performed for multi-user rc2(1M)
 streamio(7) STREAMS ioctl commands performed to stop the rc0(1M)
 version(8) version commands streamio(7)
 ar(4) common archive file format ar(4)
 output a.out(4) common assembler and link editor a.out(4)
 ldfcn(4) common object file access routines ldfcn(4)
 linenum(4) line number entries in a common object file linenum(4)
 relocation information for a common object file reloc(4) reloc(4)
 scnhdr(4) section header for a common object file scnhdr(4)
 format syms(4) common object file symbol table syms(4)
 filehdr(4) file header for a common object files filehdr(4)
 descriptions infocmp(1M) compare or print out terminfo infocmp(1M)
 regex(5) regular expression compile and match routines regex(5)
 term(4) format of compiled term file term(4)
 tic(1M) terminfo compiler tic(1M)
 3B2boot(8) 3B2 computer bootstrap procedures 3B2boot(8)

Permuted Index

if(7) 3B2 computer Floppy Disk Subsystem if(7)
id(7) 3B2 computer Integral Disk Subsystem id(7)
makes a floppy key for the 3B2 Computer newkey(8) newkey(8)
 master(4) master configuration database master(4)
 system(4) system configuration information table system(4)
prtconf(1M) print system configuration prtconf(1M)
 lpadmin(1M) configure the LP print service lpadmin(1M)
 baud(8) display or change console baud rate baud(8)
 console(7) console interface console(7)
 console(7) console interface console(7)
header for implementation-specific constants limits(4) file limits(4)
 math(5) math functions and constants math(5)
 unistd(4) file header for symbolic constants unistd(4)
 mkfs(1M) construct a file system mkfs(1M)
 disks makehdfs(1M) construct file systems on hard makehdfs(1M)
debugging on Uutry(1M) try to contact remote system with Uutry(1M)
 init(1M) telinit(1M) process control initialization init(1M)
 jagent(5) host control of windowing terminal jagent(5)
 fcntl(5) file control options fcntl(5)
 uadmin(1M) administrative control uadmin(1M)
 tty(7) controlling terminal interface tty(7)
 term(5) conventional names for terminals term(5)
 character classification and conversion tables /generate chrtrbl(1M)
 a terminfo/ captinfo(1M) convert a termcap description into captinfo(1M)
bootable object file mkboot(1M) convert an object file to a mkboot(1M)
 dd(1M) convert and copy a file dd(1M)
one-password file/ pwunconv(1M) Converts from a two- to a pwunconv(1M)
timod(7) Transport Interface cooperating STREAMS module timod(7)
 dd(1M) convert and copy a file dd(1M)
 cartridge tape ctccpio(1M) copy file archives in and out to ctccpio(1M)
 access time dcopy(1M) copy file systems for optimal dcopy(1M)
 volcopy(1M) make literal copy of file system volcopy(1M)
 core(4) format of core image file core(4)
 mem(7) kmem(7) core memory mem(7)
 core(4) format of core image file core(4)
 cpio(4) format of cpio archive cpio(4)
 cpio(4) format of cpio archive cpio(4)
 crash(1M) examine system images crash(1M)
 makefsys(1M) create a file system on a diskette makefsys(1M)
 command gencc(1M) create a front-end to the cc gencc(1M)
inittab entries for ports/ ports(8) create character device files and ports(8)
 cron(1M) clock daemon cron(1M)
 ctccpio(1M) copy file archives in ctccpio(1M)
 ctcfmt(1M) format cartridge tape ctcfmt(1M)
 ctcinfo(1M) display information ctcinfo(1M)
 current layer relogin(1M) relogin(1M)
 curses screen image file scr_dump(4)
 daemon cron(1M)
Error status report command and Log Daemon hdelogger(1M) Hard Disk hdelogger(1M)

rfudaemon(1M) Remote File Sharing
 strerr(1M) STREAMS error logger
 terminfo(4) terminal capability
 stat(5)
 types(5) primitive system
 master(4) master configuration
 optimal access time
 fsdb(1M) file system
 try to contact remote system with
 timezone(4) set
 sysdef(1M) output system
 termcap description into a terminfo
 captainfo(1M) convert a termcap
 compare or print out terminfo
 fstyp(1M)
 Equipped Device/ disks(1M) adds
 for/ ports(8) create character
 bad block mapping on a hard disk
 devnm(1M)
 clone(7) open any minor
 print the VTOC of a block
 devinfo(1M) print
 edt(8) Equipped
 for hard disks in the Equipped
 filledt(8) fill Equipped
 Administration SA(7)
 print major number(s) of hardware
 information
 blocks and i-nodes
 systems fsck(1M)
 firmware
 dgmon(8) run
 uuclean(1M) check the uucp
 dir(4) format of
 link and unlink files and
 uucleanup(1M) uucp spool
 dirent(4) file system independent
 chroot(1M) change root
 adv(1M) advertise a
 mvdire(1M) move a
 directory entry
 type, modes, speed, and line
 type, modes, speed, and line
 fusage(1M)
 sadb(1M)
 df(1M) report number of free

daemon process rfudaemon(1M)
 daemon strerr(1M)
 data base terminfo(4)
 data returned by stat system call stat(5)
 data types types(5)
 database master(4)
 dcopy(1M) copy file systems for dcopy(1M)
 dd(1M) convert and copy a file dd(1M)
 debugger fsdb(1M)
 debugging on Uutry(1M) Uutry(1M)
 default system time zone timezone(4)
 definition sysdef(1M)
 description /convert a captainfo(1M)
 description into a terminfo/ captainfo(1M)
 descriptions infocmp(1M) infocmp(1M)
 determine file system identifier fstyp(1M)
 /dev/entries for hard disks in the disks(1M)
 device files and inittab entries ports(8)
 device hdefix(1) report or change hdefix(1)
 device name devnm(1M)
 device on a STREAMS driver clone(7)
 device prtvtoc(1M) prtvtoc(1M)
 device specific information devinfo(1M)
 Device Table commands edt(8)
 Device Table /adds /dev/entries disks(1M)
 Device Table (EDT) filledt(8)
 devices administered by System SA(7)
 devices getmajor(1M) getmajor(1M)
 devinfo(1M) print device specific devinfo(1M)
 devnm(1M) device name devnm(1M)
 df(1M) report number of free disk df(1M)
 dfscck(1M) check and repair file fsck(1M)
 dgmon(8) run diagnostic phases in dgmon(8)
 diagnostic phases in firmware dgmon(8)
 dir(4) format of directories dir(4)
 directories and permissions file uuclean(1M)
 directories dir(4)
 directories link(1M) unlink(1M) link(1M)
 directory clean-up uucleanup(1M)
 directory entry dirent(4)
 directory for a command chroot(1M)
 directory for remote access adv(1M)
 directory mvdire(1M)
 dirent(4) file system independent dirent(4)
 discipline getty(1M) set terminal getty(1M)
 discipline /set terminal uugetty(1M)
 disk access profiler fusage(1M)
 disk access profiler sadb(1M)
 disk blocks and i-nodes df(1M)

Permuted Index

load olboot and mboot onto the
check a file system on a removable
change bad block mapping on a hard
 hdelog(7) hard
hdeadd(1M) add/delete hdelog (Hard
and Log Daemon hdelogger(1M) Hard
 physically format a SCSI hard
 id(7) 3B2 computer Integral
 if(7) 3B2 computer Floppy
 du(1M) summarize
umountfsys(1M) mount, unmount a
 create a file system on a
 fmtflop(1M) physically format
load system dump from floppy
fmthard(1M) populate VTOC on hard
 /adds /dev/entries for hard
construct file systems on hard
dump system memory image to floppy
hard disks in the Equipped Device/
 tape ctctinfo(1M)
 information rmntstat(1M)
 baud(8)
object downloader for the 5620
domain and network names
 whodo(1M) who is
dname(1M) Print Remote File Sharing
peripheral board pump(1M)
terminal wtinit(1M) object
open any minor device on a STREAMS
 drvinstall(1M) install/uninstall a
 xt(7) multiplexed tty
 xtd(1M) extract and print xt
 xtd(1M) extract and print xt
 xts(1M) extract and print xt
 sxt(7) pseudo-device
 system file with kernel and
 channels protocol used by xt(7)
 driver
 ldsysdump(1M) load system
disk(s) sysdump(8) boot option to
 edittbl(1M)
a.out(4) common assembler and link
 software application file
 fill Equipped Device Table
 commands
 edittbl(1M) edit
 application file
disk boot partition newboot(1M) newboot(1M)
disk checkfsys(1M) checkfsys(1M)
disk device hdefix(1) report or hdefix(1)
disk error log interface file hdelog(7)
Disk Error Log) reports hdeadd(1M)
Disk Error status report command hdelogger(1M)
disk format(1M) format(1M)
Disk Subsystem id(7)
Disk Subsystem if(7)
disk usage du(1M)
diskette file system mountfsys(1M) mountfsys(1M)
diskette makefsys(1M) makefsys(1M)
diskettes fmtflop(1M)
diskettes ldsysdump(1M) ldsysdump(1M)
disks fmthard(1M)
disks in the Equipped Device Table disks(1M)
disks makehdfs(1M) makehdfs(1M)
disk(s) sysdump(8) boot option to sysdump(8)
disks(1M) adds /dev/entries for disks(1M)
display information about cartridge ctctinfo(1M)
display mounted resource rmntstat(1M)
display or change console baud rate baud(8)
DMD terminal wtinit(1M) wtinit(1M)
dname(1M) Print Remote File Sharing dname(1M)
doing what whodo(1M)
domain and network names dname(1M)
Download B16 or X86 a.out file to a pump(1M)
downloader for the 5620 DMD wtinit(1M)
driver clone(7) clone(7)
driver drvinstall(1M)
driver for AT&T windowing terminals xt(7)
driver link structure xtd(1M)
driver packet traces xtd(1M)
driver statistics xts(1M)
driver sxt(7)
driver symbol tables /a bootable mkunix(1M)
driver xtproto(5) multiplexed xtproto(5)
drvinstall(1M) install/uninstall a drvinstall(1M)
du(1M) summarize disk usage du(1M)
dump from floppy diskettes ldsysdump(1M)
dump system memory image to floppy sysdump(8)
edit edt_data file edittbl(1M)
editor output a.out(4)
editsa(1M) add/delete entry from editsa(1M)
edittbl(1M) edit edt_data file edittbl(1M)
(EDT) filledt(8) filledt(8)
edt(8) Equipped Device Table edt(8)
edt_data file edittbl(1M)
.edt_swapp(4) softwareedt_swapp(4)

character device files and inittab
 linenum(4) line number
 file system independent directory
 utmp(4) wtmp(4) utmp and wtmp
 file editsa(1M) add/delete
 relogin(1M) rename login
 profile(4) setting up an
 environ(5) user
 commands performed for multi-user
 stop the Remote File Sharing
 edt(8)
 /dev/entries for hard disks in the
 filledt(8) fill
 errdump(1M) print
 hdelog(7) hard disk
 add/delete hdelog (Hard Disk
 strclean(1M) STREAMS
 strerr(1M) STREAMS
 log(7) interface to STREAMS
 Daemon hdelogger(1M) Hard Disk
 setmnt(1M)
 to STREAMS error logging and
 crash(1M)
 uuxqt(1M)
 routines regexp(5) regular
 structure xtd(1M)
 traces xtt(1M)
 statistics xts(1M)
 pathalias(4) alias file for
 finc(1M)
 statistics for a file system
 ldfcn(4) common object
 cartridge tape ctccpio(1M) copy
 pwck(1M) grpck(1M) password/group
 fcntl(5)
 core(4) format of core image
 dd(1M) convert and copy a
 entry from software application
 edittbl(1M) edit edt_data
 software application
 pathalias(4) alias
 acct(4) per-process accounting
 ar(4) common archive
 pnch(4)
 intro(4) introduction to
 group(4) group
 entries for ports boards /create ports(8)
 entries in a common object file linenum(4)
 entry dirent(4) dirent(4)
 entry formats utmp(4)
 entry from software application editsa(1M)
 entry to show current layer relogin(1M)
 environ(5) user environment environ(5)
 environment at login time profile(4)
 environment environ(5)
 environment rc2(1M) run rc2(1M)
 environment rfstop(1M) rfstop(1M)
 Equipped Device Table commands edt(8)
 Equipped Device Table /adds disks(1M)
 Equipped Device Table (EDT) filledt(8)
 errdump(1M) print error log errdump(1M)
 error log errdump(1M)
 error log interface file hdelog(7)
 Error Log reports hdeadd(1M) hdeadd(1M)
 error logger cleanup program strclean(1M)
 error logger daemon strerr(1M)
 error logging and event tracing log(7)
 Error status report command and Log hdelogger(1M)
 establish mount table setmnt(1M)
 event tracing log(7) interface log(7)
 examine system images crash(1M)
 execute remote command requests uuxqt(1M)
 expression compile and match regexp(5)
 extract and print xt driver link xtd(1M)
 extract and print xt driver packet xtt(1M)
 extract and print xt driver xts(1M)
 FACE pathalias(4)
 fast incremental backup finc(1M)
 fcntl(5) file control options fcntl(5)
 ff(1M) list file names and ff(1M)
 file access routines ldfcn(4)
 file archives in and out to ctccpio(1M)
 file checkers pwck(1M)
 file control options fcntl(5)
 file core(4)
 file dd(1M)
 file editsa(1M) add/delete editsa(1M)
 file edittbl(1M)
 file .edt_swapp(4)edt_swapp(4)
 file for FACE pathalias(4)
 file format acct(4)
 file format ar(4)
 file format for card images pnch(4)
 file formats intro(4)
 file group(4)

Permuted Index

hard disk error log interface
 filehdr(4)
implementation-specific/
 limits(4)
 unistd(4)
 issue(4) issue identification
number entries in a common object
 an object file to a bootable object
 mknod(1M) build special
 file system ff(1M) list
 null(7) the null
 identify processes using a
 passwd(4) password
information for a common object
File Sharing name server master
 scsfile(4) format of SCCS
 from a two- to a one-password
section header for a common object
 format of curses screen image
 rfadmin(1M) Remote
 rfudaemon(1M) Remote
names dname(1M) Print Remote
 rfstop(1M) stop the Remote
 rfpasswd(1M) change Remote
 file rfmaster(4) Remote
 nsquery(1M) Remote
 script rfuadmin(1M) Remote
unadv(1M) unadvertise a Remote
 /mount, unmount Remote
 rfstart(1M) start Remote
 idload(1M) Remote
identify processes using a file or
 syms(4) common object
 ckbupscd(1M) check
 fsba(1M)
 fsdb(1M)
file names and statistics for a
 fstyp(1M) determine
 entry dirent(4)
 mkfs(1M) construct a
mount, unmount a diskette
 makefsys(1M) create a
 checkfsys(1M) check a
 fsstat(1M) report
 mnttab(4) mounted
 cmpress(1M) re-link
volcopy(1M) make literal copy of
 volume fs(4)
/umount(1M) mount and unmount
 time dcopy(1M) copy
file hdelog(7)
file header for common object files
file header for
file header for symbolic constants
file
file linenum(4) line
file mkboot(1M) convert
file
file names and statistics for a
file
file or file structure fuser(1M)
file
file reloc(4) relocation
file rfmaster(4) Remote
file
file scheme pwunconv(1M) Converts
file scnhdr(4)
file scr_dump(4)
File Sharing administration
File Sharing daemon process
File Sharing domain and network
File Sharing environment
File Sharing host password
File Sharing name server master
File Sharing name server query
File Sharing notification shell
File Sharing resource
File Sharing resources
File Sharing
File Sharing user and group mapping
file structure fuser(1M)
file symbol table format
file system backup schedule
file system block analyzer
file system debugger
file system ff(1M) list
file system identifier
file system independent directory
file system
file system /umountfsys(1M)
file system on a diskette
file system on a removable disk
file system status
file system table
file system to remove fragmentation
file system
file system(4) format of system
file systems and remote resources
file systems for optimal access

fsck(1M) dfscck(1M) check and repair
 labelit(1M) provide labels for
 mount, unmount multiple
 makehdfs(1M) construct
 ncheck checklist(4) list of
 term(4) format of compiled term
 mkboot(1M) convert an object
 pump(1M) Download B16 or X86 a.out
 system uucico(1M)
 the scheduler for the uucp
 uucp directories and permissions
 mkunix(1M) make a bootable system
 object files
 link(1M) unlink(1M) link and unlink
 ports(8) create character device
 file header for common object
 frec(1M) recover
 format specification in text
 string, format of graphical
 intro(7) introduction to special
 passgmt(1M) password
 information .ott(4)
 fstab(4)
 filledt(8)
 Table (EDT)
 service lpfiler(1M) administer
 reconfigured at boot/ ckauto(1M)
 dgmon(8) run diagnostic phases in
 passwd(8) change
 led(1M)
 set NVRAM parameters for
 if(7) 3B2 computer
 ldsysdump(1M) load system dump from
 to dump system memory image to
 newkey(8) makes a
 for floating boot
 diskettes
 disks
 resource fumount(1M)
 format(1M) physically
 acct(4) per-process accounting file
 ar(4) common archive file
 ctcfmt(1M)
 fmtflop(1M) physically
 pnch(4) file
 inode(4)
 term(4)
 core(4)

file systems fsck(1M)
 file systems labelit(1M)
 file systems /umountall(1M) mountall(1M)
 file systems on hard disks makehdfs(1M)
 file systems processed by fsck and checklist(4)
 file term(4)
 file to a bootable object file mkboot(1M)
 file to a peripheral board pump(1M)
 file transport program for the uucp uucico(1M)
 file transport program uusched(1M) uusched(1M)
 file uucheck(1M) check the uucheck(1M)
 file with kernel and driver symbol/ mkunix(1M)
 filehdr(4) file header for common filehdr(4)
 files and directories link(1M)
 files and inittab entries for ports/ ports(8)
 files filehdr(4) filehdr(4)
 files from a backup tape frec(1M)
 files fspec(4) fspec(4)
 files gps(4) graphical primitive gps(4)
 files intro(7)
 files management passgmt(1M)
 files that hold object architecture &.ott(4)
 file-system-table fstab(4)
 fill Equipped Device Table (EDT) filledt(8)
 filledt(8) fill Equipped Device filledt(8)
 filters used with the LP print lpfiler(1M)
 func(1M) fast incremental backup func(1M)
 find if the UNIX system was ckauto(1M)
 firmware dgmon(8)
 firmware password passwd(8)
 flash green LED led(1M)
 floating boot fltboot(1M) fltboot(1M)
 Floppy Disk Subsystem if(7)
 floppy diskettes ldsysdump(1M)
 floppy disk(s) /boot option sysdump(8)
 floppy key for the 3B2 Computer newkey(8)
 fltboot(1M) set NVRAM parameters fltboot(1M)
 fmtflop(1M) physically format fmtflop(1M)
 fmthard(1M) populate VTOC on hard fmthard(1M)
 forced unmount of an advertised fumount(1M)
 format a SCSI hard disk format(1M)
 format acct(4)
 format ar(4)
 format cartridge tape ctcfmt(1M)
 format diskettes fmtflop(1M)
 format for card images pnch(4)
 format of an i-node inode(4)
 format of compiled term file term(4)
 format of core image file core(4)

Permuted Index

cpio(4) format of cpio archive cpio(4)
scr_dump(4) format of curses screen image file scr_dump(4)
dir(4) format of directories dir(4)
gps(4) graphical primitive string, gps(4)
format of graphical files gps(4)
format of SCCS file sccsfile(4)
fs(4) file system(4) format of system volume fs(4)
fspec(4) format specification in text files fspec(4)
common object file symbol table format syms(4) syms(4)
hard disk format(1M) physically format a SCSI format(1M)
formats intro(4)
intro(4) introduction to file formats utmp(4)
utmp(4) utmp and wtmp entry forms used with the LP print lpforms(1M)
service lpforms(1M) administer fragmentation cmpress(1M) cmpress(1M)
re-link file system to remove backup tape freq(1M) recover files from a freq(1M)
df(1M) report number of free disk blocks and i-nodes df(1M)
gencc(1M) create a front-end to the cc command gencc(1M)
system volume fs(4) file system(4) format of fs(4)
fsba(1M) file system block analyzer fsba(1M)
list of file systems processed by fsck and ncheck checklist(4) checklist(4)
file systems fsck(1M) dfck(1M) check and repair fsck(1M)
fsdb(1M) file system debugger fsdb(1M)
text files fspec(4) format specification in fspec(4)
status fsstat(1M) report file system fsstat(1M)
fstab(4) file-system-table fstab(4)
identifier fstyp(1M) determine file system fstyp(1M)
advertised resource fumount(1M) forced unmount of an fumount(1M)
prof(5) profile within a function prof(5)
math(5) math functions and constants math(5)
fusage(1M) disk access profiler fusage(1M)
a file or file structure fuser(1M) identify processes using fuser(1M)
cc command gencc(1M) create a front-end to the gencc(1M)
termio(7) general terminal interface termio(7)
and conversion tables chrtbl(1M) generate character classification chrtbl(1M)
ncheck(1M) generate path names from i-numbers ncheck(1M)
of hardware devices getmajor(1M) print major number(s) getmajor(1M)
speed and terminal settings used by getty gettydefs(4) gettydefs(4)
speed, and line discipline getty(1M) set terminal type, modes, getty(1M)
settings used by getty gettydefs(4) speed and terminal gettydefs(4)
format of graphical files gps(4) graphical primitive string, gps(4)
primitive string, format of graphical files gps(4) graphical gps(4)
of graphical files gps(4) graphical primitive string, format gps(4)
plot(4) graphics interface plot(4)
led(1M) flash green LED led(1M)
group(4) group file group(4)
id(1M) print user and group IDs and names id(1M)
Remote File Sharing user and group mapping idload(1M) idload(1M)
newgrp(1M) log in to a new group newgrp(1M)
group(4) group file group(4)
checkers pwck(1M) grpck(1M) password/group file pwck(1M)

varargs(5) handle variable argument list varargs(5)
 or change bad block mapping on a
 hdelog(7) hard disk device hdefix(1) report hdefix(1)
 hdelog(7) hard disk error log interface file hdelog(7)
 hdeadd(1M) add/delete hdelog (Hard Disk Error Log) reports hdeadd(1M)
 command and Log/ hdelogger(1M) Hard Disk Error status report hdelogger(1M)
 format(1M) physically format a SCSI hard disk format(1M)
 fmthard(1M) populate VTOC on hard disks fmthard(1M)
 disks(1M) adds /dev/entries for hard disks in the Equipped Device/ disks(1M)
 construct file systems on hard disks makehdfs(1M) makehdfs(1M)
 setclk(1M) set system time from hardware clock setclk(1M)
 print major number(s) of hardware devices getmajor(1M) getmajor(1M)
 Disk Error Log) reports hdeadd(1M) add/delete hdelog (Hard hdeadd(1M)
 block mapping on a hard disk/ hdefix(1) report or change bad hdefix(1)
 reports hdeadd(1M) add/delete hdelog (Hard Disk Error Log) hdeadd(1M)
 interface file hdelog(7) hard disk error log hdelog(7)
 status report command and Log/ hdelogger(1M) Hard Disk Error hdelogger(1M)
 scnhdr(4) section header for a common object file scnhdr(4)
 filehdr(4) file header for common object files filehdr(4)
 constants limits(4) file header for implementation-specific limits(4)
 unistd(4) file header for symbolic constants unistd(4)
 information .ott(4) files that hold object architecture &.ott(4)
 layers(5) protocol used between host and windowing terminal layers(5)
 jagent(5) host control of windowing terminal jagent(5)
 change Remote File Sharing host password rfpasswd(1M) rfpasswd(1M)
 names id(1M) print user and group IDs and id(1M)
 Subsystem id(7) 3B2 computer Integral Disk id(7)
 issue(4) issue identification file issue(4)
 fstyp(1M) determine file system identifier fstyp(1M)
 file structure fuser(1M) identify processes using a file or fuser(1M)
 and group mapping idload(1M) Remote File Sharing user idload(1M)
 id(1M) print user and group IDs and names id(1M)
 Subsystem if(7) 3B2 computer Floppy Disk if(7)
 core(4) format of core image file core(4)
 scr_dump(4) format of curses screen image file scr_dump(4)
 boot option to dump system memory image to floppy disk(s) sysdump(8) sysdump(8)
 crash(1M) examine system images crash(1M)
 pnch(4) file format for card images pnch(4)
 limits(4) file header for implementation-specific constants limits(4)
 finc(1M) fast incremental backup finc(1M)
 dirent(4) file system independent directory entry dirent(4)
 terminfo descriptions infocmp(1M) compare or print out infocmp(1M)
 ctcinfo(1M) display information about cartridge tape ctcinfo(1M)
 devinfo(1M) print device specific information devinfo(1M)
 file reloc(4) relocation information for a common object reloc(4)
 files that hold object architecture information .ott(4) &.ott(4)
 display mounted resource information rmntstat(1M) rmntstat(1M)
 system(4) system configuration information table system(4)
 inittab(4) script for the init process inittab(4)
 control initialization init(1M) telinit(1M) process init(1M)

Permuted Index

telinit(1M) process control	initialization init(1M)	init(1M)
brc(1M) bcheckrc(1M) system	initialization procedures	brc(1M)
/create character device files and	inittab entries for ports boards	ports(8)
process	inittab(4) script for the init	inittab(4)
cli(1M) clear	i-node	cli(1M)
inode(4) format of an	i-node	inode(4)
number of free disk blocks and	inode(4) format of an i-node	inode(4)
install(1M)	i-nodes df(1M) report	df(1M)
drvinstall(1M)	install commands	install(1M)
id(7) 3B2 computer	install(1M) install commands	install(1M)
console(7) console	install/uninstall a driver	drvinstall(1M)
module timod(7) Transport	Integral Disk Subsystem	id(7)
hdelog(7) hard disk error log	interface	console(7)
mt(7) tape	Interface cooperating STREAMS	timod(7)
plot(4) graphics	interface file	hdelog(7)
ports(7) 5 line asynchronous	interface	mt(7)
STREAMS module tirdwr(7) Transport	interface	plot(4)
Transport Interface read/write	interface	ports(7)
swap(1M) swap administrative	Interface read/write interface	tirdwr(7)
termio(7) general terminal	interface STREAMS module tirdwr(7)	tirdwr(7)
administration sysadm(1) menu	interface	swap(1M)
and event tracing log(7)	interface	termio(7)
tty(7) controlling terminal	interface to do system	sysadm(1)
maintenance commands and/	interface to STREAMS error logging	log(7)
formats	interface	tty(7)
files	intro(1M) introduction to	intro(1M)
maintenance procedures	intro(4) introduction to file	intro(4)
intro(4)	intro(5) introduction to miscellany	intro(5)
commands and application/	intro(7) introduction to special	intro(7)
intro(1M)	intro(8) introduction to system	intro(8)
intro(5)	introduction to file formats	intro(4)
intro(7)	introduction to maintenance	intro(1M)
procedures intro(8)	introduction to miscellany	intro(5)
ncheck(1M) generate path names from	introduction to special files	intro(7)
streamio(7) STREAMS	introduction to system maintenance	intro(8)
issue(4)	i-numbers	ncheck(1M)
terminal	ioctl commands	streamio(7)
/make a bootable system file with	issue identification file	issue(4)
newkey(8) makes a floppy	issue(4) issue identification file	issue(4)
killall(1M)	jagent(5) host control of windowing	jagent(5)
processes	kernel and driver symbol tables	mkunix(1M)
mem(7)	key for the 3B2 Computer	newkey(8)
systems	kill all active processes	killall(1M)
labelit(1M) provide	killall(1M) kill all active	killall(1M)
cftime(4)	kmem(7) core memory	mem(7)
rename login entry to show current	labelit(1M) provide labels for file	labelit(1M)
	labels for file systems	labelit(1M)
	language specific strings	cftime(4)
	layer relogin(1M)	relogin(1M)

host and windowing terminal under routines
 floppy diskettes
 led(1M) flash green

implementation-specific constants
 ports(7) 5

terminal type, modes, speed, and
 terminal type, modes, speed, and
 object file linenum(4)
 common object file
 directories link(1M) unlink(1M)
 a.out(4) common assembler and
 xtd(1M) extract and print xt driver
 files and directories
 a file system ff(1M)
 fsck and ncheck checklist(4)

varargs(5) handle variable argument
 nlsadmin(1M) network
 volcopy(1M) make
 boot partition newboot(1M)
 diskettes ldsysdump(1M)

Error status report command and
 errdump(1M) print error
 newgrp(1M)
 hdelog(7) hard disk error

add/delete hdelog (Hard Disk Error
 logging, and event tracing
 strclean(1M) STREAMS error
 strerr(1M) STREAMS error

log(7) interface to STREAMS error
 relogin(1M) rename
 setting up an environment at
 /lpmove(1M) start/stop the
 lpadmin(1M) configure the
 administer filters used with the
 administer forms used with the
 reject(1M) allow or prevent
 service
 used with the LP print service
 with the LP print service

service and/ lpshut(1M) lpshut(1M)
 start/stop the LP print service/
 the LP print service/ lpshut(1M)
 priorities
 values(5)
 intro(1M) introduction to
 intro(8) introduction to system
 getmajor(1M) print

layers(5) protocol used between layers(5)
 ldfcn(4) common object file access ldfcn(4)
 ldsysdump(1M) load system dump from ... ldsysdump(1M)
 LED led(1M)
 led(1M) flash green LED led(1M)
 limits(4) file header for limits(4)
 line asynchronous interface ports(7)
 line discipline getty(1M) set getty(1M)
 line discipline ugetty(1M) set ugetty(1M)
 line number entries in a common linenum(4)
 linenum(4) line number entries in a linenum(4)
 link and unlink files and link(1M)
 link editor output a.out(4)
 link structure xtd(1M)
 link(1M) unlink(1M) link and unlink link(1M)
 list file names and statistics for ff(1M)
 list of file systems processed by checklist(4)
 list varargs(5)
 listener service administration nlsadmin(1M)
 literal copy of file system volcopy(1M)
 load olboot and mboot onto the disk newboot(1M)
 load system dump from floppy ldsysdump(1M)
 Log Daemon hdelogger(1M) Hard Disk hdelogger(1M)
 log errdump(1M)
 log in to a new group newgrp(1M)
 log interface file hdelog(7)
 Log reports hdeadd(1M) hdeadd(1M)
 log(7) interface to STREAMS error log(7)
 logger cleanup program strclean(1M)
 logger daemon strerr(1M)
 logging and event tracing log(7)
 login entry to show current layer relogin(1M)
 login time profile(4) profile(4)
 LP print service and move requests lpsched(1M)
 LP print service lpadmin(1M)
 LP print service lpfilter(1M) lpfilter(1M)
 LP print service lpforms(1M) lpforms(1M)
 LP requests accept(1M) accept(1M)
 lpadmin(1M) configure the LP print lpadmin(1M)
 lpfilter(1M) administer filters lpfilter(1M)
 lpforms(1M) administer forms used lpforms(1M)
 lpmove(1M) start/stop the LP print lpsched(1M)
 lpsched(1M) lpshut(1M) lpmove(1M) lpsched(1M)
 lpshut(1M) lpmove(1M) start/stop lpsched(1M)
 lpusers(1M) set printing queue lpusers(1M)
 machine-dependent values values(5)
 maintenance commands and/ intro(1M)
 maintenance procedures intro(8)
 major number(s) of hardware devices getmajor(1M)

rc2(1M) run commands performed for	multi-user environment	rc2(1M)
devnm(1M) device	mvdir(1M) move a directory	mvdir(1M)
rfmaster(4) Remote File Sharing	name	devnm(1M)
nsquery(1M) Remote File Sharing	name server master file	rfmaster(4)
system ff(1M) list file	name server query	nsquery(1M)
File Sharing domain and network	names and statistics for a file	ff(1M)
term(5) conventional	names dname(1M) Print Remote	dname(1M)
ncheck(1M) generate path	names for terminals	term(5)
id(1M) print user and group IDs and	names from i-numbers	ncheck(1M)
file systems processed by fsck and	names	id(1M)
i-numbers	ncheck checklist(4) list of	checklist(4)
administration nlsadmin(1M)	ncheck(1M) generate path names from	ncheck(1M)
Remote File Sharing domain and	network listener service	nlsadmin(1M)
onto the disk boot partition	network names dname(1M) Print	dname(1M)
the 3B2 Computer	newboot(1M) load olboot and mboot	newboot(1M)
service administration	newgrp(1M) log in to a new group	newgrp(1M)
rfuadmin(1M) Remote File Sharing	newkey(8) makes a floppy key for	newkey(8)
name server query	nlsadmin(1M) network listener	nlsadmin(1M)
null(7) the	notification shell script	rfuadmin(1M)
file linenum(4) line	nsquery(1M) Remote File Sharing	nsquery(1M)
i-nodes df(1M) report	null file	null(7)
getmajor(1M) print major	null(7) the null file	null(7)
fltboot(1M) set	number entries in a common object	linenum(4)
.ott(4) files that hold	number of free disk blocks and	df(1M)
terminal wtinit(1M)	number(s) of hardware devices	getmajor(1M)
ldfcn(4) common	NVRAM parameters for floating boot	fltboot(1M)
line number entries in a common	object architecture information	&.ott(4)
an object file to a bootable	object downloader for the 5620 DMD	wtinit(1M)
relocation information for a common	object file access routines	ldfcn(4)
section header for a common	object file linenum(4)	linenum(4)
syms(4) common	object file mkboot(1M) convert	mkboot(1M)
file mkboot(1M) convert an	object file reloc(4)	reloc(4)
filehdr(4) file header for common	object file scnhdr(4)	scnhdr(4)
partition newboot(1M) load	object file symbol table format	syms(4)
/Converts from a two- to a	object file to a bootable object	mkboot(1M)
newboot(1M) load olboot and mboot	object files	filehdr(4)
driver clone(7)	olboot and mboot onto the disk boot	newboot(1M)
prf(7)	one-password file scheme	pwunconv(1M)
run commands performed to stop the	onto the disk boot partition	newboot(1M)
dcopy(1M) copy file systems for	open any minor device on a STREAMS	clone(7)
to floppy disk(s) sysdump(8) boot	operating system profiler	prf(7)
fcntl(5) file control	operating system rc0(1M)	rc0(1M)
architecture information	optimal access time	dcopy(1M)
common assembler and link editor	option to dump system memory image	sysdump(8)
sysdef(1M)	options	fcntl(5)
sadc(1M) system activity report	.ott(4) files that hold object	&.ott(4)
	output a.out(4)	a.out(4)
	output system definition	sysdef(1M)
	package sar(1M) sa1(1M) sa2(1M)	sar(1M)

Permuted Index

xtt(1M) extract and print xt driver
 fltboot(1M) set NVRAM
olboot and mboot onto the disk boot
 management
 passwd(4)
 passmgmt(1M)
 passwd(8) change firmware
change Remote File Sharing host
 pwck(1M) grpck(1M)
 ncheck(1M) generate
environment rc2(1M) run commands
 system rc0(1M) run commands
Download B16 or X86 a.out file to a
 check the uucp directories and
 acct(4)
 dgmon(8) run diagnostic
 format(1M)
 fmtflop(1M)
 fmthard(1M)
 files and inittab entries for
 interface
 files and inittab entries for/
stop all processes and turn off the
 and turn off the power
 accept(1M) reject(1M) allow or
 profiler(1M) prfld(1M) prfstat(1M)
prfsnap(1M) prfpr(1M)/ profiler(1M)
/prfstat(1M) prfdc(1M) prfsnap(1M)
/prfld(1M) prfstat(1M) prfdc(1M)
prfpr(1M)/ profiler(1M) prfld(1M)
 graphical files gps(4) graphical
 types(5)
 devinfo(1M)
 errdump(1M)
 devices getmajor(1M)
 infocmp(1M) compare or
and network names dname(1M)
 /lpmove(1M) start/stop the LP
 lpadmin(1M) configure the LP
administer filters used with the LP
administer forms used with the LP
 strace(1M)
 prtconf(1M)
packet traces xtt(1M)
parameters for floating boot fltboot(1M)
partition newboot(1M) load newboot(1M)
passmgmt(1M) password files passmgmt(1M)
passwd(4) password file passwd(4)
passwd(8) change firmware password passwd(8)
password file passwd(4)
password files management passmgmt(1M)
password passwd(8)
password rfpaswd(1M) rfpaswd(1M)
password/group file checkers pwck(1M)
path names from i-numbers ncheck(1M)
pathalias(4) alias file for FACE pathalias(4)
performed for multi-user rc2(1M)
performed to stop the operating rc0(1M)
peripheral board pump(1M) pump(1M)
permissions file uucheck(1M) uucheck(1M)
per-process accounting file format acct(4)
phases in firmware dgmon(8)
physically format a SCSI hard disk format(1M)
physically format diskettes fmtflop(1M)
plot(4) graphics interface plot(4)
pnch(4) file format for card images pnch(4)
populate VTOC on hard disks fmthard(1M)
ports boards /character device ports(8)
ports(7) 5 line asynchronous ports(7)
ports(8) create character device ports(8)
power powerdown(1M) powerdown(1M)
powerdown(1M) stop all processes powerdown(1M)
prevent LP requests accept(1M)
prf(7) operating system profiler prf(7)
prfdc(1M) prfsnap(1M) prfpr(1M)/ profiler(1M)
prfld(1M) prfstat(1M) prfdc(1M) profiler(1M)
prfpr(1M) UNIX system profiler profiler(1M)
prfsnap(1M) prfpr(1M) UNIX system/ profiler(1M)
prfstat(1M) prfdc(1M) prfsnap(1M) profiler(1M)
primitive string, format of gps(4)
primitive system data types types(5)
print device specific information devinfo(1M)
print error log errdump(1M)
print major number(s) of hardware getmajor(1M)
print out terminfo descriptions infocmp(1M)
Print Remote File Sharing domain dname(1M)
print service and move requests lpsched(1M)
print service lpadmin(1M)
print service lpfilter(1M) lpfilter(1M)
print service lpforms(1M) lpforms(1M)
print STREAMS trace messages strace(1M)
print system configuration prtconf(1M)

prtvtoc(1M) print the VTOC of a block device prtvtoc(1M)
 id(1M) print user and group IDs and names id(1M)
 xtd(1M) extract and print xt driver link structure xtd(1M)
 xtt(1M) extract and print xt driver packet traces xtt(1M)
 xts(1M) extract and print xt driver statistics xts(1M)
 lpusers(1M) set printing queue priorities lpusers(1M)
 lpusers(1M) set printing queue priorities lpusers(1M)
 3B2boot(8) 3B2 computer bootstrap procedures 3B2boot(8)
 bcheckrc(1M) system initialization procedures brc(1M) brc(1M)
 introduction to system maintenance procedures intro(8) intro(8)
 init(1M) telinit(1M) process control initialization init(1M)
 inittab(4) script for the init process inittab(4)
 Remote File Sharing daemon process rfudaemon(1M) rfudaemon(1M)
 checklist(4) list of file systems processed by fsck and ncheck checklist(4)
 powerdown(1M) stop all processes and turn off the power powerdown(1M)
 killall(1M) kill all active processes killall(1M)
 structure fuser(1M) identify processes using a file or file fuser(1M)
 prof(5) profile within a function prof(5)
 profile within a function prof(5)
 environment at login time profile(4) setting up an profile(4)
 fusage(1M) disk access profiler fusage(1M)
 prf(7) operating system profiler prf(7)
 prfsnap(1M) prfpr(1M) UNIX system profiler /prfstat(1M) prfdc(1M) profiler(1M)
 sadb(1M) disk access profiler sadp(1M)
 prfdc(1M) prfsnap(1M) prfpr(1M)/ profiler(1M) prfld(1M) prfstat(1M) profiler(1M)
 uucico(1M) file transport program for the uucp system uucico(1M)
 STREAMS error logger cleanup program strclean(1M) strclean(1M)
 for the uucp file transport program uusched(1M) the scheduler uusched(1M)
 commands and application programs /to maintenance intro(1M)
 windowing terminal under layers(5) protocol used between host and layers(5)
 xtproto(5) multiplexed channels protocol used by xt(7) driver xtproto(5)
 labelit(1M) provide labels for file systems labelit(1M)
 configuration prtconf(1M) print system prtconf(1M)
 block device prtvtoc(1M) print the VTOC of a prtvtoc(1M)
 sxt(7) pseudo-device driver sxt(7)
 file to a peripheral board pump(1M) Download B16 or X86 a.out pump(1M)
 file checkers pwck(1M) grpck(1M) password/group pwck(1M)
 to a one-password file scheme pwunconv(1M) Converts from a two- pwunconv(1M)
 Remote File Sharing name server query nsquery(1M) nsquery(1M)
 lpusers(1M) set printing queue priorities lpusers(1M)
 rmount(1M) queue remote resource mounts rmount(1M)
 rumount(1M) cancel queued remote resource request rumount(1M)
 rmntry(1M) attempt to mount queued remote resources rmntry(1M)
 display or change console baud rate baud(8) baud(8)
 stop the operating system rc0(1M) run commands performed to rc0(1M)
 multi-user environment rc2(1M) run commands performed for rc2(1M)
 tirdwr(7) Transport Interface read/write interface STREAMS module tirdwr(7)
 /find if the UNIX system was reconfigured at boot time ckauto(1M)
 frec(1M) recover files from a backup tape frec(1M)

Permuted Index

compile and match routines
 match routines `regex(5)`
 requests `accept(1M)`
fragmentation `cmpress(1M)`
 a common object file
 object file `reloc(4)`
 show current layer
commands from source code `mk(8)`
 `adv(1M)` advertise a directory for
 uxqt(1M) execute
 rfadmin(1M)
 rfudaemon(1M)
network names `dnname(1M)` Print
 rfstop(1M) stop the
 rfpasswd(1M) change
 master file `rfmaster(4)`
 query `nsquery(1M)`
 shell script `rfuadmin(1M)`
 unadv(1M) unadvertise a
/rumountall(1M) mount, unmount
 rfstart(1M) start
 mapping `idload(1M)`
 rmount(1M) queue
 rumount(1M) cancel queued
mount and unmount file systems and
`rmnttry(1M)` attempt to mount queued
 Uutry(1M) try to contact
 check a file system on a
`cmpress(1M)` re-link file system to
 layer `relogin(1M)`
`fsck(1M)` `dfscck(1M)` check and
 /Hard Disk Error status
 fsstat(1M)
 and i-nodes `df(1M)`
 on a hard disk device `hdefix(1)`
`sa2(1M)` `sadc(1M)` system activity
 hdelog (Hard Disk Error Log)
 cancel queued remote resource
 reject(1M) allow or prevent LP
 the LP print service and move
`uuxqt(1M)` execute remote command
 forced unmount of an advertised
 rmntstat(1M) display mounted
 rmount(1M) queue remote
`rumount(1M)` cancel queued remote
 unadvertise a Remote File Sharing
and unmount file systems and remote
 attempt to mount queued remote
mount, unmount Remote File Sharing
 regex(5) regular expression
 regular expression compile and regex(5)
 reject(1M) allow or prevent LP
 re-link file system to remove cmpress(1M)
`reloc(4)` relocation information for reloc(4)
 relocation information for a common reloc(4)
`relogin(1M)` rename login entry to relogin(1M)
 remake the binary system and mk(8)
 remote access adv(1M)
 remote command requests uuxqt(1M)
Remote File Sharing administration rfadmin(1M)
Remote File Sharing daemon process rfudaemon(1M)
Remote File Sharing domain and dnname(1M)
Remote File Sharing environment rfstop(1M)
Remote File Sharing host password rfpasswd(1M)
Remote File Sharing name server rfmaster(4)
Remote File Sharing name server nsquery(1M)
Remote File Sharing notification rfuadmin(1M)
Remote File Sharing resource unadv(1M)
Remote File Sharing resources rmountall(1M)
Remote File Sharing rfstart(1M)
Remote File Sharing user and group idload(1M)
remote resource mounts rmount(1M)
remote resource request rumount(1M)
remote resources /umount(1M) mount(1M)
remote resources rmnttry(1M)
remote system with debugging on Uutry(1M)
removable disk `checkfsys(1M)` checkfsys(1M)
remove fragmentation cmpress(1M)
rename login entry to show current relogin(1M)
repair file systems fsck(1M)
report command and Log Daemon hdelogger(1M)
report file system status fsstat(1M)
report number of free disk blocks df(1M)
report or change bad block mapping hdefix(1)
report package `sa(1M)` `sa1(1M)` sa(1M)
reports `hdeadd(1M)` add/delete hdeadd(1M)
request `rumount(1M)` rumount(1M)
requests `accept(1M)` accept(1M)
requests /lpmove(1M) start/stop lpsched(1M)
requests uuxqt(1M)
resource `fumount(1M)` fumount(1M)
resource information rmntstat(1M)
resource mounts rmount(1M)
resource request rumount(1M)
resource unadv(1M) unadv(1M)
resources /umount(1M) mount mount(1M)
resources `rmnttry(1M)` rmnttry(1M)
resources /rumountall(1M) rmountall(1M)

stat(5) data returned by stat system call stat(5)
 administration rfadmin(1M) Remote File Sharing rfadmin(1M)
 name server master file rfmaster(4) Remote File Sharing rfmaster(4)
 Sharing host password rfpasswd(1M) change Remote File rfpasswd(1M)
 Sharing rfstart(1M) start Remote File rfstart(1M)
 Sharing environment rfstop(1M) stop the Remote File rfstop(1M)
 notification shell script rfuadmin(1M) Remote File Sharing rfuadmin(1M)
 daemon process rfudaemon(1M) Remote File Sharing rfudaemon(1M)
 resource information rmntstat(1M) display mounted rmntstat(1M)
 remote resources rmntry(1M) attempt to mount queued rmntry(1M)
 mounts rmount(1M) queue remote resource rmount(1M)
 unmount Remote File Sharing/ rmountall(1M) rumountall(1M) mount, rmountall(1M)
 chroot(1M) change root directory for a command chroot(1M)
 ldfcn(4) common object file access routines ldfcn(4)
 expression compile and match routines regexp(5) regular regexp(5)
 resource request rumount(1M) cancel queued remote rumount(1M)
 Remote File Sharing/ rmountall(1M) rumountall(1M) mount, unmount rmountall(1M)
 multi-user environment rc2(1M) run commands performed for rc2(1M)
 operating system rc0(1M) run commands performed to stop the rc0(1M)
 dgmon(8) run diagnostic phases in firmware dgmon(8)
 activity report package sar(1M) sa1(1M) sa2(1M) sadc(1M) system sar(1M)
 report package sar(1M) sa1(1M) sa2(1M) sadc(1M) system activity sar(1M)
 System Administration SA(7) devices administered by SA(7)
 package sar(1M) sa1(1M) sa2(1M) sadc(1M) system activity report sar(1M)
 sadp(1M) disk access profiler sadp(1M)
 sar(1M) sa1(1M) sa2(1M) sadc(1M) sar(1M)
 system activity report package SCCS file sccsfile(4)
 sccsfile(4) format of SCCS file sccsfile(4)
 check file system backup schedule ckbupscd(1M) ckbupscd(1M)
 transport program uusched(1M) the scheduler for the uucp file uusched(1M)
 from a two- to a one-password file scheme pwunconv(1M) Converts pwunconv(1M)
 common object file scnhdr(4) section header for a scnhdr(4)
 image file scr_dump(4) format of curses screen scr_dump(4)
 scr_dump(4) format of curses screen image file scr_dump(4)
 inittab(4) script for the init process inittab(4)
 File Sharing notification shell script rfuadmin(1M) Remote rfuadmin(1M)
 format(1M) physically format a SCSI hard disk format(1M)
 file scnhdr(4) section header for a common object scnhdr(4)
 Remote File Sharing name server master file rfmaster(4) rfmaster(4)
 Remote File Sharing name server query nsquery(1M) nsquery(1M)
 nlsadmin(1M) network listener service administration nlsadmin(1M)
 /lpmove(1M) start/stop the LP print service and move requests lpsched(1M)
 lpadmin(1M) configure the LP print service lpadmin(1M)
 filters used with the LP print service lpfilter(1M) administer lpfilter(1M)
 forms used with the LP print service lpforms(1M) administer lpforms(1M)
 ascii(5) map of ASCII character set ascii(5)
 timezone(4) set default system time zone timezone(4)
 boot fltboot(1M) set NVRAM parameters for floating fltboot(1M)
 lpusers(1M) set printing queue priorities lpusers(1M)

Permuted Index

setclk(1M) set system time from hardware clock setclk(1M)
and line discipline getty(1M) set terminal type, modes, speed, getty(1M)
and line discipline uugetty(1M) set terminal type, modes, speed, uugetty(1M)
hardware clock setclk(1M) set system time from setclk(1M)
setmnt(1M) establish mount table setmnt(1M)
setting up an environment at login profile(4)
settings used by getty gettydefs(4)
Sharing administration rfadmin(1M)
Sharing daemon process rfudaemon(1M)
Sharing domain and network names dname(1M)
Sharing environment rfstop(1M)
Sharing host password rfpasswd(1M)
Sharing name server master file rfmaster(4)
Sharing name server query nsquery(1M)
Sharing notification shell script rfuadmin(1M)
Sharing resource unadv(1M)
Sharing resources /rmountall(1M) rmountall(1M)
Sharing rfstart(1M)
Sharing user and group mapping idload(1M)
shell script rfuadmin(1M) rfuadmin(1M)
show current layer relogin(1M)
shut down system, change system shutdown(1M)
shutdown(1M) shut down system, shutdown(1M)
software application file editisa(1M)
software application fileedt_swapp(4)
source code mk(8) remake mk(8)
special file mknod(1M)
special files intro(7)
specific information devinfo(1M)
specific strings cftime(4)
specification in text files fspec(4)
speed, and line discipline getty(1M)
speed, and line discipline uugetty(1M)
speed and terminal settings used by gettydefs(4)
pool directory clean-up ucleanup(1M)
start Remote File Sharing rfstart(1M)
start/stop the LP print service and/ lpsched(1M)
stat system call stat(5)
stat(5) data returned by stat stat(5)
statistics for a file system ff(1M)
statistics xts(1M)
status fsstat(1M)
status report command and Log/ hdelogger(1M)
stop all processes and turn off the powerdown(1M)
stop the operating system rc0(1M)
stop the Remote File Sharing rfstop(1M)
strace(1M) print STREAMS trace strace(1M)
strclean(1M) STREAMS error logger strclean(1M)
streamio(7) STREAMS ioctl commands streamio(7)

clone(7) open any minor device on a program
 strclean(1M) STREAMS driver clone(7)
 strerr(1M) STREAMS error logger cleanup strclean(1M)
 tracing log(7) interface to STREAMS error logger daemon strerr(1M)
 streamio(7) STREAMS error logging and event log(7)
 Transport Interface cooperating STREAMS ioctl commands streamio(7)
 Interface read/write interface STREAMS module timod(7) timod(7)
 strace(1M) print STREAMS module tirdwr(7) Transport tirdwr(7)
 daemon STREAMS trace messages strace(1M)
 gps(4) graphical primitive strerr(1M) STREAMS error logger strerr(1M)
 cftime(4) language specific string, format of graphical files gps(4)
 processes using a file or file strings cftime(4)
 extract and print xt driver link structure fuser(1M) identify fuser(1M)
 user structure xtd(1M) xtd(1M)
 id(7) 3B2 computer Integral Disk su(1M) become super-user or another su(1M)
 if(7) 3B2 computer Floppy Disk Subsystem id(7)
 du(1M) Subsystem if(7)
 sync(1M) update the summarize disk usage du(1M)
 su(1M) become super block sync(1M)
 swap(1M) interface super-user or another user su(1M)
 swap(1M) swap administrative interface swap(1M)
 swap(1M) swap administrative swap(1M) swap administrative swap(1M)
 sxt(7) pseudo-device driver sxt(7)
 syms(4) common object file symbol table format syms(4)
 system file with kernel and driver symbol tables /make a bootable mkunix(1M)
 unistd(4) file header for symbolic constants unistd(4)
 table format syms(4) common object file symbol syms(4)
 system administration sync(1M) update the super block sync(1M)
 sysadm(1) menu interface to do sysadm(1)
 sysdef(1M) output system definition sysdef(1M)
 sysdump(8) boot option to dump sysdump(8)
 system activity report package sar(1M)
 System Administration SA(7)
 system administration sysadm(1)
 system and commands from source mk(8)
 system backup schedule ckbupscd(1M)
 system block analyzer fsba(1M)
 system call stat(5)
 system, change system state shutdown(1M)
 system configuration information system(4)
 system configuration prtconf(1M)
 system data types types(5)
 system debugger fsdb(1M)
 system definition sysdef(1M)
 system dump from floppy diskettes ldsysdump(1M)
 system ff(1M) list file ff(1M)
 system file with kernel and driver mkunix(1M)
 system identifier fstyp(1M)
 system images crash(1M)
 system independent directory entry dirent(4)

Permuted Index

brc(1M) bcheckrc(1M) system initialization procedures brc(1M)
intro(8) introduction to system maintenance procedures intro(8)
sysdump(8) boot option to dump system memory image to floppy/ sysdump(8)
mkfs(1M) construct a file system mkfs(1M)
mount, unmount a diskette system /umountfsys(1M) mountfsys(1M)
makefsys(1M) create a file system on a diskette makefsys(1M)
checkfsys(1M) check a file system on a removable disk checkfsys(1M)
prf(7) operating system profiler prf(7)
prfsnap(1M) prfpr(1M) UNIX system profiler /prfdc(1M) profiler(1M)
performed to stop the operating system rc0(1M) run commands rc0(1M)
shut down system, change system state shutdown(1M) shutdown(1M)
fsstat(1M) report file system status fsstat(1M)
mnttab(4) mounted file system table mnttab(4)
setclk(1M) set system time from hardware clock setclk(1M)
timezone(4) set default system time zone timezone(4)
cmpress(1M) re-link file system to remove fragmentation cmpress(1M)
file transport program for the uucp system uucico(1M) uucico(1M)
make literal copy of file system volcopy(1M) volcopy(1M)
fs(4) file system(4) format of system volume fs(4)
time ckauto(1M) find if the UNIX system was reconfigured at boot ckauto(1M)
Uutry(1M) try to contact remote system with debugging on Uutry(1M)
fs(4) file system(4) format of system volume fs(4)
information table system(4) system configuration system(4)
/umount(1M) mount and unmount file systems and remote resources mount(1M)
dcopy(1M) copy file systems for optimal access time dcopy(1M)
dfsc(1M) check and repair file systems fsck(1M) fsck(1M)
labelit(1M) provide labels for file systems labelit(1M)
mount, unmount multiple file systems mountall(1M) umountall(1M) mountall(1M)
makehdfs(1M) construct file systems on hard disks makehdfs(1M)
ncheck checklist(4) list of file systems processed by fsck and checklist(4)
edt(8) Equipped Device Table commands edt(8)
hard disks in the Equipped Device Table /adds /dev/entries for disks(1M)
filledt(8) fill Equipped Device Table (EDT) filledt(8)
syms(4) common object file symbol table format syms(4)
mnttab(4) mounted file system table mnttab(4)
setmnt(1M) establish mount table setmnt(1M)
system configuration information table system(4) system(4)
classification and conversion tables /generate character chrtbl(1M)
file with kernel and driver symbol tables /make a bootable system mkunix(1M)
archives in and out to cartridge tape ctccpio(1M) copy file ctccpio(1M)
ctcfmt(1M) format cartridge tape ctcfmt(1M)
display information about cartridge tape ctinfo(1M) ctinfo(1M)
recover files from a backup tape frec(1M) frec(1M)
mt(7) tape interface mt(7)
initialization init(1M) telinit(1M) process control init(1M)
term(4) format of compiled term file term(4)
file term(4) format of compiled term term(4)
terminals term(5) conventional names for term(5)
captainfo(1M) convert a termcap description into a terminfo/ captainfo(1M)

terminfo(4) terminal capability data base terminfo(4)
 termio(7) general terminal interface termio(7)
 tty(7) controlling terminal interface tty(7)
 jagent(5) host control of windowing terminal jagent(5)
 gettydefs(4) speed and terminal settings used by getty gettydefs(4)
 line discipline getty(1M) set terminal type, modes, speed, and getty(1M)
 line discipline uugetty(1M) set terminal type, modes, speed, and uugetty(1M)
 used between host and windowing terminal under layers(5) protocol layers(5)
 object downloader for the 5620 DMD terminal wtinit(1M) wtinit(1M)
 term(5) conventional names for terminals term(5)
 tty driver for AT&T windowing terminals xt(7) multiplexed xt(7)
 tic(1M) terminfo compiler tic(1M)
 a termcap description into a terminfo description /convert captainfo(1M)
 infocmp(1M) compare or print out terminfo descriptions infocmp(1M)
 data base terminfo(4) terminal capability terminfo(4)
 interface termio(7) general terminal termio(7)
 fspec(4) format specification in text files fspec(4)
 zone tic(1M) terminfo compiler tic(1M)
 cooperating STREAMS module timezone(4) set default system time timezone(4)
 read/write interface STREAMS/ timod(7) Transport Interface timod(7)
 strace(1M) print STREAMS tirdwr(7) Transport Interface tirdwr(7)
 extract and print xt driver packet trace messages strace(1M)
 to STREAMS error logging and event traces xtt(1M) xtt(1M)
 STREAMS module timod(7) tracing log(7) interface log(7)
 interface STREAMS module tirdwr(7) Transport Interface cooperating timod(7)
 system uucico(1M) file Transport Interface read/write tirdwr(7)
 the scheduler for the uucp file transport program for the uucp uucico(1M)
 debugging on Uutry(1M) transport program usched(1M) usched(1M)
 terminals xt(7) multiplexed try to contact remote system with Uutry(1M)
 interface tty driver for AT&T windowing xt(7)
 stop all processes and tty(7) controlling terminal tty(7)
 pwunconv(1M) Converts from a turn off the power powerdown(1M) powerdown(1M)
 discipline getty(1M) set terminal two- to a one-password file scheme pwunconv(1M)
 uugetty(1M) set terminal type, modes, speed, and line getty(1M)
 types(5) primitive system data type, modes, speed, and line/ uugetty(1M)
 types types(5) primitive system data types(5)
 types(5) primitive system data types(5)
 uadmin(1M) administrative control uadmin(1M)
 umount(1M) mount and unmount file mount(1M)
 umountall(1M) mount, unmount mountall(1M)
 mountfsys(1M) mount, unmount a mountfsys(1M)
 unadv(1M) unadvertise a Remote File unadv(1M)
 unadvertise a Remote File Sharing unadv(1M)
 unistd(4) file header for symbolic unistd(4)
 UNIX system profiler /prfstat(1M) profiler(1M)
 UNIX system was reconfigured at ckauto(1M)
 unlink files and directories link(1M)
 unlink(1M) link and unlink files link(1M)
 unmount a diskette file system mountfsys(1M)

Permuted Index

mount(1M) umount(1M) mount and
mountall(1M) umountall(1M) mount,
 fumount(1M) forced
rmountall(1M) rumountall(1M) mount,
 sync(1M)
 du(1M) summarize disk
 id(1M) print
idload(1M) Remote File Sharing
 environ(5)
su(1M) become super-user or another
 fuser(1M) identify processes
 utmp(4) wtmp(4)
 formats
directories and permissions file
 for the uucp system
 clean-up
 file ucheck(1M) check the
usched(1M) the scheduler for the
 uucleanup(1M)
 file transport program for the
modes, speed, and line discipline
 uucp file transport program
 system with debugging on
 requests
values(5) machine-dependent
 list
 varargs(5) handle
 version(8)
 file system
file system(4) format of system
 prvtoc(1M) print the
 fmthard(1M) populate
 jagent(5) host control of
protocol used between host and
multiplexed tty driver for AT&T
 prof(5) profile
 the 5620 DMD terminal
 utmp(4) wtmp(4) utmp and
 utmp(4)
board pump(1M) Download B16 or
 xtd(1M) extract and print
 xtt(1M) extract and print
 xts(1M) extract and print
channels protocol used by
AT&T windowing terminals
 link structure
 unmount file systems and remote/ mount(1M)
 unmount multiple file systems mountall(1M)
 unmount of an advertised resource fumount(1M)
 unmount Remote File Sharing/ rmountall(1M)
 update the super block sync(1M)
 usage du(1M)
 user and group IDs and names id(1M)
 user and group mapping idload(1M)
 user environment environ(5)
 user su(1M)
 using a file or file structure fuser(1M)
 utmp and wtmp entry formats utmp(4)
 utmp(4) wtmp(4) utmp and wtmp entry utmp(4)
 uucheck(1M) check the uucp uuccheck(1M)
 uucico(1M) file transport program uucico(1M)
 uucleanup(1M) uucp spool directory uucleanup(1M)
 uucp directories and permissions uuccheck(1M)
 uucp file transport program uusched(1M)
 uucp spool directory clean-up uucleanup(1M)
 uucp system uucico(1M) uucico(1M)
 uugetty(1M) set terminal type, uugetty(1M)
 uusched(1M) the scheduler for the uusched(1M)
Uutry(1M) try to contact remote Uutry(1M)
uuxqt(1M) execute remote command uuxqt(1M)
values values(5)
values(5) machine-dependent values values(5)
varargs(5) handle variable argument varargs(5)
variable argument list varargs(5)
version commands version(8)
version(8) version commands version(8)
volcopy(1M) make literal copy of volcopy(1M)
volume fs(4) fs(4)
VTOC of a block device prvtoc(1M)
VTOC on hard disks fmthard(1M)
whodo(1M) who is doing what whodo(1M)
windowing terminal jagent(5)
windowing terminal under layers(5) layers(5)
windowing terminals xt(7) xt(7)
within a function prof(5)
wtinit(1M) object downloader for wtinit(1M)
wtmp entry formats utmp(4)
wtmp(4) utmp and wtmp entry formats utmp(4)
X86 a.out file to a peripheral pump(1M)
xt driver link structure xtd(1M)
xt driver packet traces xtt(1M)
xt driver statistics xts(1M)
xt(7) driver /multiplexed xtproto(5)
xt(7) multiplexed tty driver for xt(7)
xtd(1M) extract and print xt driver xtd(1M)



NAME

intro – introduction to maintenance commands and application programs

DESCRIPTION

This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes. The commands in this section should be used along with those listed in Section 1 of the *User's Reference Manual* and Sections 1, 2, 3, 4, and 5 of the *Programmer's Reference Manual*. References of the form *name*(1), (2), (3), (4) and (5) refer to entries in the above manuals. References of the form *name*(1M), *name*(7) or *name*(8) refer to entries in this manual.

COMMAND SYNTAX

Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [*option*(s)] [*cmdarg*(s)]

where:

name The name of an executable file.

option – *noargletter*(s) or,
– *argletter* <> *optarg*
where <> is optional white space.

noargletter A single letter representing an option without an argument.

argletter A single letter representing an option requiring an argument.

optarg Argument (character string) satisfying preceding *argletter*.

cmdarg Path name (or other command argument) *not* beginning with – or, – by itself indicating the standard input.

SEE ALSO

getopt(1) in the *User's Reference Manual*.

getopt(3C) in the *Programmer's Reference Manual*.

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of "normal" termination) one supplied by the program (see *wait*(2) and *exit*(2)). The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and non-zero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously "exit code", "exit status", or "return code", and is described only where special conventions are involved.

BUGS

Regrettably, not all commands adhere to the aforementioned syntax.



NAME

accept, reject – allow or prevent LP requests

SYNOPSIS

`/usr/lib/accept destinations`
`/usr/lib/reject [-r[reason]] destinations`

DESCRIPTION

accept allows *lp(1)* to accept requests for the named *destinations*. A *destination* can be either a line printer (LP) or a class of printers. Use *lpstat(1)* to find the status of *destinations*.

Reject prevents *lp(1)* from accepting requests for the named *destinations*. A *destination* can be either a printer or a class of printers. Use *lpstat(1)* to find the status of *destinations*. The following option is useful with *reject*.

`-r[reason]` Associates a *reason* with preventing *lp* from accepting requests. This *reason* applies to all printers mentioned up to the next `-r` option. *Reason* is reported by *lp* when users direct requests to the named *destinations* and by *lpstat(1)*. If the `-r` option is not present or the `-r` option is given without a *reason*, then a default *reason* will be used.

FILES

`/usr/spool/lp/*`

SEE ALSO

lpadmin(1M), *lpsched(1M)*,
enable(1), *lp(1)*, *lpstat(1)* in the *User's Reference Manual*.



NAME

adv – advertise a directory for remote access

SYNOPSIS

```
adv [-r] [-d description] resource pathname [clients...]
adv -m resource -d description | [clients...]
adv -m resource [-d description] | clients...
adv
```

DESCRIPTION

adv is the Remote File Sharing command used to make a resource from one computer available for use on other computers. The machine that advertises the resource is called the *server*, while computers that mount and use the resource are *clients*. [See **mount(1M)**.] (A resource represents a directory, which could contain files, subdirectories, named pipes and devices.)

There are three ways **adv** is used: 1) to advertise the directory *pathname* under the name *resource* so it is available to Remote File Sharing *clients*; 2) to modify *client* and *description* fields for currently advertised resources; or 3) to print a list of all locally-advertised resources.

The following options are available:

- r** Restricts access to the resource to a read-only basis. The default is read-write access.
- d *description*** Provides brief textual information about the advertised resource. *description* is a single argument surrounded by double quotes (") and has a maximum length of 32 characters.
- resource*** This is the symbolic name used by the server and all authorized clients to identify the resource. It is limited to a maximum of 14 characters and must be different from every other resource name in the domain. All characters must be printable ASCII characters but must not include periods (.), slashes (/), or white space.
- pathname*** This is the local pathname of the advertised resource. It is limited to a maximum of 64 characters. This pathname cannot be the mount point of a remote resource and it can only be advertised under one resource name.
- clients*** These are the names of all clients that are authorized to remotely mount the resource. The default is that all machines that can connect to the server are authorized to access the resource. Valid input is of the form *nodename*, *domain.nodename*, *domain.*, or an alias that represents a list of client names. A domain name must be followed by a period (.) to distinguish it from a host name. The aliases are defined in **/etc/host.alias** and must conform to the alias capability in **mailx(1)**.
- m** This option modifies information for a resource that has already been advertised. The resource is identified by a *resource* name. Only the *clients* and *description* fields can be

modified. (To change the *pathname*, *resource* name, or read/write permissions, you must unadvertise and re-advertise the resource.)

When used with no options, *adv* displays all local resources that have been advertised; this includes the resource name, the *pathname*, the description, the read-write status, and the list of authorized clients. The resource field has a fixed length of 14 characters; all others are of variable length. Fields are separated by two white spaces, double quotes (") surround the description, and blank lines separate each resource entry.

This command may be used without options by any user; otherwise it is restricted to the super-user.

Remote File Sharing must be running before *adv* can be used to advertise or modify a resource entry.

EXIT STATUS

If there is at least one syntactically valid entry in the *clients* field, a warning will be issued for each invalid entry and the command will return a successful exit status. A non-zero exit status will be returned if the command fails.

ERRORS

If (1) the network is not up and running, (2) *pathname* is not a directory, (3) *pathname* isn't on a file system mounted locally, or (4) there is at least one entry in the *clients* field but none are syntactically valid, an error message will be sent to standard error.

FILES

/etc/host.alias

SEE ALSO

mount(1M), *rfstart(1M)*, *unadv(1M)*.
mailx(1) in the *User's Reference Manual*.

NAME

brc, *bcheckrc* – system initialization procedures

SYNOPSIS

/etc/brc

/etc/bcheckrc

DESCRIPTION

These shell procedures are executed via entries in */etc/inittab* by *init*(1M) whenever the system is booted (or rebooted).

First, the *bcheckrc* procedure checks the status of the root file system. If the root file system is found to be bad, *bcheckrc* repairs it.

Then, the *brc* procedure clears the mounted file system table, */etc/mnttab* and puts the entry for the root file system into the mount table.

After these two procedures have executed, *init* checks for the *initdefault* value in */etc/inittab*. This tells *init* in which run level to place the system. Since *initdefault* is initially set to **2**, the system will be placed in the multi-user state via the */etc/rc2* procedure.

Note that *bcheckrc* should always be executed before *brc*. Also, these shell procedures may be used for several run-level states.

SEE ALSO

fsck(1M), *init*(1M), *rc2*(1M), *shutdown*(1M).



NAME

captoinfo – convert a termcap description into a terminfo description

SYNOPSIS

captoinfo [-v ...] [-V] [-1] [-w width] file ...

DESCRIPTION

captoinfo looks in *file* for *termcap* descriptions. For each one found, an equivalent *terminfo*(4) description is written to standard output, along with any comments found. A description which is expressed as relative to another description (as specified in the *termcap* *tc=* field) will be reduced to the minimum superset before being output.

If no *file* is given, then the environment variable *TERMCAP* is used for the filename or entry. If *TERMCAP* is a full pathname to a file, only the terminal whose name is specified in the environment variable *TERM* is extracted from that file. If the environment variable *TERMCAP* is not set, then the file */etc/termcap* is read.

- v print out tracing information on standard error as the program runs. Specifying additional -v options will cause more detailed information to be printed.
- V print out the version of the program in use on standard error and exit.
- 1 cause the fields to print out one to a line. Otherwise, the fields will be printed several to a line to a maximum width of 60 characters.
- w change the output to *width* characters.

FILES

*/usr/lib/terminfo/?/** compiled terminal description database

CAVEATS

Certain *termcap* defaults are assumed to be true. For example, the bell character (*terminfo* *bel*) is assumed to be ^G . The linefeed capability (*termcap* *nl*) is assumed to be the same for both *cursor_down* and *scroll_forward* (*terminfo* *cu**l* and *ind*, respectively.) Padding information is assumed to belong at the end of the string.

The algorithm used to expand parameterized information for *termcap* fields such as *cursor_position* (*termcap* *cm*, *terminfo* *cup*) will sometimes produce a string which, though technically correct, may not be optimal. In particular, the rarely used *termcap* operation *%n* will produce strings that are especially long. Most occurrences of these non-optimal strings will be flagged with a warning message and may need to be recoded by hand.

The short two-letter name at the beginning of the list of names in a *termcap* entry, a hold-over from an earlier version of the UNIX system, has been removed.

DIAGNOSTICS

tgetent failed with return code *n* (reason).

The *termcap* entry is not valid. In particular, check for an invalid '*tc=*' entry.

unknown type given for the *termcap* code *cc*.

The *termcap* description had an entry for *cc* whose type was not boolean, numeric or string.

wrong type given for the boolean (numeric, string) *termcap* code *cc*.

The boolean *termcap* entry *cc* was entered as a numeric or string capability.

the boolean (numeric, string) *termcap* code *cc* is not a valid name.

An unknown *termcap* code was specified.

tgetent failed on *TERM=term*.

The terminal type specified could not be found in the *termcap* file.

TERM=term: cap cc (info ii) is NULL: REMOVED

The *termcap* code was specified as a null string. The correct way to cancel an entry is with an '@', as in ':bs@:'. Giving a null string could cause incorrect assumptions to be made by the software which uses *termcap* or *terminfo*.

a function key for *cc* was specified, but it already has the value *vv*.

When parsing the *ko* capability, the key *cc* was specified as having the same value as the capability *cc*, but the key *cc* already had a value assigned to it.

the unknown *termcap* name *cc* was specified in the *ko* *termcap* capability.

A key was specified in the *ko* capability which could not be handled.

the *vi* character *v* (*info ii*) has the value *xx*, but *ma* gives *n*.

The *ma* capability specified a function key with a value different from that specified in another setting of the same key.

the unknown *vi* key *v* was specified in the *ma* *termcap* capability.

A *vi(1)* key unknown to *captoinfo* was specified in the *ma* capability.

Warning: *termcap sg (nn)* and *termcap ug (nn)* had different values.

terminfo assumes that the *sg* (now *xmc*) and *ug* values were the same.

Warning: the string produced for *ii* may be inefficient.

The parameterized string being created should be rewritten by hand.

Null *termname* given.

The terminal type was null. This is given if the environment variable *TERM* is not set or is null.

cannot open *file* for reading.

The specified file could not be opened.

SEE ALSO

infocmp(1M), tic(1M), terminfo(4).
curses (3X) in the *Programmer's Reference Manual*.
Chapter 10 in the *Programmer's Guide*.

NOTES

captoinfo should be used to convert *termcap* entries to *terminfo(4)* entries because the *termcap* database (from earlier versions of UNIX System V) may not be supplied in future releases.



NAME

checkfsys – check a file system on a removable disk

SYNOPSIS

The *checkfsys* command allows the user to check for and optionally repair a damaged file system on a removable disk.

The user is asked one of the following three functions:

check the file system

No repairs are attempted.

repair it interactively

The user is informed about each instance of damage and asked if it should be repaired.

repair it automatically

The program applies a standard repair to each instance of damage.

The identical function is available under the *sysadm* menu:

sysadm checkfsys

The command may be assigned a password. See *sysadm(1)*, the **admpasswd** sub-command.

WARNING

While automatic and interactive checks are generally successful, they can occasionally lose a file or a file's name. Files with content but without names are put in the */file-system/lost+found* directory.

If losing data is of particular concern, "check" the file system first to discover if it appears to be damaged. If it is damaged, use one of the repair mechanisms or the file system debugging utility, **fsdb**.

SEE ALSO

fsck(1M), *fsdb(1M)*, *makefsys(1M)*, *mountfsys(1M)*.
sysadm(1) in the *User's Reference Manual*.



NAME

chroot – change root directory for a command

SYNOPSIS

```
/etc/chroot newroot command
```

DESCRIPTION

chroot causes the given command to be executed relative to the new root. The meaning of any initial slashes (/) in the path names is changed for the command and any of its child processes to *newroot*. Furthermore, upon execution, the initial working directory is *newroot*.

Notice, however, that if you redirect the output of the command to a file:

```
chroot newroot command >x
```

will create the file *x* relative to the original root of the command, not the new one.

The new root path name is always relative to the current root: even if a *chroot* is currently in effect, the *newroot* argument is relative to the current root of the running process.

This command can be run only by the super-user.

SEE ALSO

cd(1) in the *User's Reference Manual*.

chroot(2) in the *Programmer's Reference Manual*.

BUGS

One should exercise extreme caution when referencing device files in the new root file system.



NAME

`chrtbl` – generate character classification and conversion tables

SYNOPSIS

`chrtbl` [*file*]

DESCRIPTION

The `chrtbl` command creates a character classification table and an upper/lower-case conversion table. The tables are contained in a byte-sized array encoded such that a table lookup can be used to determine the character classification of a character or to convert a character (see `ctype(3C)`). The size of the array is 257*2 bytes: 257 bytes are required for the 8-bit code set character classification table and 257 bytes for the upper- to lower-case and lower- to upper-case conversion table.

`chrtbl` reads the user-defined character classification and conversion information from *file* and creates two output files in the current directory. One output file, `ctype.c` (a C-language source file), contains the 257*2-byte array generated from processing the information from *file*. You should review the content of `ctype.c` to verify that the array is set up as you had planned. (In addition, an application program could use `ctype.c`.) The first 257 bytes of the array in `ctype.c` are used for character classification. The characters used for initializing these bytes of the array represent character classifications that are defined in `/usr/include/ctype.h`; for example, `_L` means a character is lower case and `_S|_B` means the character is both a spacing character and a blank. The last 257 bytes of the array are used for character conversion. These bytes of the array are initialized so that characters for which you do not provide conversion information will be converted to themselves. When you do provide conversion information, the first value of the pair is stored where the second one would be stored normally, and vice versa; for example, if you provide `<0x41 0x61>`, then `0x61` is stored where `0x41` would be stored normally, and `0x41` is stored where `0x61` would be stored normally.

The second output file (a data file) contains the same information, but is structured for efficient use by the character classification and conversion routines (see `ctype(3C)`). The name of this output file is the value of the character classification `chrclass` read in from *file*. This output file must be installed in the `/lib/chrclass` directory under this name by someone who is super-user or a member of group `bin`. This file must be readable by user, group, and other; no other permissions should be set. To use the character classification and conversion tables on this file, set the environmental variable `CHRCLASS` (see `environ(5)`) to the name of this file and export the variable; for example, if the name of this file (and character class) is `xyz`, you should issue the commands: `CHRCLASS=xyz ; export CHRCLASS`.

If no input file is given, or if the argument `-` is encountered, `chrtbl` reads from the standard input file.

The syntax of *file* allows the user to define the name of the data file created by `chrtbl`, the assignment of characters to character classifications and the relationship between upper- and lower-case letters. The character classifications recognized by `chrtbl` are:

chrclass	name of the data file to be created by <i>chrtbl</i> .
isupper	character codes to be classified as upper-case letters.
islower	character codes to be classified as lower-case letters.
isdigit	character codes to be classified as numeric.
isspace	character codes to be classified as a spacing (delimiter) character.
ispunct	character codes to be classified as a punctuation character.
iscntrl	character codes to be classified as a control character.
isblank	character code for the space character.
isxdigit	character codes to be classified as hexadecimal digits.
ul	relationship between upper- and lower-case characters.

Any lines with the number sign (#) in the first column are treated as comments and are ignored. Blank lines are also ignored.

A character can be represented as a hexadecimal or octal constant (for example, the letter a can be represented as 0x61 in hexadecimal or 0141 in octal). Hexadecimal and octal constants may be separated by one or more space and tab characters.

The dash character (-) may be used to indicate a range of consecutive numbers. Zero or more space characters may be used for separating the dash character from the numbers.

The backslash character (\) is used for line continuation. Only a carriage return is permitted after the backslash character.

The relationship between upper- and lower-case letters (**ul**) is expressed as ordered pairs of octal or hexadecimal constants: *<upper-case_character lower-case_character>*. These two constants may be separated by one or more space characters. Zero or more space characters may be used for separating the angle brackets (< >) from the numbers.

EXAMPLE

The following is an example of an input file used to create the ASCII code set definition table on a file named *ascii*.

```
chrclass  ascii
isupper   0x41 - 0x5a
islower   0x61 - 0x7a
isdigit   0x30 - 0x39
isspace   0x20 0x9 - 0xd
ispunct   0x21 - 0x2f 0x3a - 0x40  \
          0x5b - 0x60 0x7b - 0x7e
iscntrl   0x0 - 0x1f 0x7f
isblank   0x20
isxdigit  0x30 - 0x39 0x61 - 0x66  \
          0x41 - 0x46
```

```

u1      <0x41 0x61> <0x42 0x62> <0x43 0x63> \
        <0x44 0x64> <0x45 0x65> <0x46 0x66> \
        <0x47 0x67> <0x48 0x68> <0x49 0x69> \
        <0x4a 0x6a> <0x4b 0x6b> <0x4c 0x6c> \
        <0x4d 0x6d> <0x4e 0x6e> <0x4f 0x6f> \
        <0x50 0x70> <0x51 0x71> <0x52 0x72> \
        <0x53 0x73> <0x54 0x74> <0x55 0x75> \
        <0x56 0x76> <0x57 0x77> <0x58 0x78> \
        <0x59 0x79> <0x5a 0x7a>

```

FILES

```

/lib/chrclass/*  data file containing character classification and conversion
                  tables created by chrtbl
/usr/include/ctype.h
                  header file containing information used by character
                  classification and conversion routines

```

SEE ALSO

```

environ(5).
ctype(3C) in the Programmer's Reference Manual .

```

DIAGNOSTICS

The error messages produced by *chrtbl* are intended to be self-explanatory. They indicate errors in the command line or syntactic errors encountered within the input file.



NAME

ckauto – find if the UNIX system was reconfigured at boot time.

SYNOPSIS

/etc/ckauto

DESCRIPTION

The *ckauto* command uses a *sys3b(2)* call to determine if the UNIX system was reconfigured at the time it was last booted. The command can be used only by the super-user. If the *sys3b* call is successful (the system was reconfigured), *ckauto* exits with 1, otherwise it exits with 0. *ckauto* is intended to be used by utilities that will do services based on the output of the *sys3b* call.

SEE ALSO

sys3b(2) in the *Programmer's Reference Manual*.



NAME

ckbupscd – check file system backup schedule

SYNOPSIS

/etc/ckbupscd [-m]

DESCRIPTION

ckbupscd consults the file **/etc/bupsched** and prints the file system lists from lines with date and time specifications matching the current time. If the **-m** flag is present an introductory message in the output is suppressed so that only the file system lists are printed. Entries in the **/etc/bupsched** file are printed under the control of **cron**.

The System Administration commands *bupsched/schedcheck* are provided to review and edit the **/etc/bupsched** file.

The file **/etc/bupsched** should contain lines of 4 or more fields, separated by spaces or tabs. The first 3 fields (the schedule fields) specify a range of dates and times. The rest of the fields constitute a list of names of file systems to be printed if *ckbupscd* is run at some time within the range given by the schedule fields. The general format is:

time[time] day[,day] month[,month] fsylist

where:

time Specifies an hour of the day (0 through 23), matching any time within that hour, or an exact time of day (0:00 through 23:59).

day Specifies a day of the week (*sun* through *sat*) or day of the month (1 through 31).

month Specifies the month in which the time and day fields are valid. Legal values are the month numbers (1 through 12).

fsylist The rest of the line is taken to be a file system list to print.

Multiple time, day, and month specifications may be separated by commas, in which case they are evaluated left to right.

An asterisk (*) always matches the current value for that field.

A line beginning with a sharp sign (#) is interpreted as a comment and ignored.

The longest line allowed (including continuations) is 1024 characters.

EXAMPLES

The following are examples of lines which could appear in the **/etc/bupsched** file.

06:00-09:00 fri 1,2,3,4,5,6,7,8,9,10,11 /applic

Prints the file system name */applic* if *ckbupscd* is run between 6:00am and 9:00am any Friday during any month except December.

00:00-06:00,16:00-23:59 1,2,3,4,5,6,7 1,8 /

Prints a reminder to backup the root (/) file system if *ckbupscd* is run between the times of 4:00pm and 6:00am during the first week of August or January.

FILES

/etc/bupsched specification file containing times and file system to back up

SEE ALSO

cron(1M).

echo(1), sh(1), sysadm(1) in the *User's Reference Manual*.

BUGS

ckbupscd will report file systems due for backup if invoked any time in the window. It does not know that backups may have just been taken.

NAME

`clri` – clear i-node

SYNOPSIS

`/etc/clri special i-number ...`

DESCRIPTION

`clri` writes nulls on the 64 bytes at offset *i-number* from the start of the i-node list. This effectively eliminates the i-node at that address. *Special* is the device name on which a file system has been defined. After `clri` is executed, any blocks in the affected file will show up as “not accounted for” when `fsck(1M)` is run against the file-system. The i-node may be allocated to a new file.

Read and write permission is required on the specified *special* device.

This command is used to remove a file which appears in no directory; that is, to get rid of a file which cannot be removed with the `rm` command.

SEE ALSO

`fsck(1M)`, `fsdb(1M)`, `ncheck(1M)`, `fs(4)`.
`rm(1)` in the *User's Reference Manual*.

WARNINGS

If the file is open for writing, `clri` will not work. The file system containing the file should be NOT mounted.

If `clri` is used on the i-node number of a file that does appear in a directory, it is imperative to remove the entry in the directory at once, since the i-node may be allocated to a new file. The old directory entry, if not removed, continues to point to the same file. This sounds like a link, but does not work like one. Removing the old entry destroys the new file.



NAME

`compress` – re-link file system to remove fragmentation

SYNOPSIS

`/etc/cmpress`

DESCRIPTION

`compress` re-links the input file system to improve access time by cleaning up fragmentation of files throughout the file system. The file system must be mounted in order for this procedure to find the file system and determine its characteristics.

`compress` uses a 3B2 Computer cartridge tape for intermediate storage. The file system is first copied onto the tape, the old file system is removed and the free block list is sorted into sequential order, then the file system is copied back onto the disk so that file system blocks that previously were scattered throughout the file system are in contiguous space.

Notice that the file system is destroyed during the process of compressing it. For this reason it is strongly recommended that an up-to-date backup of the file system be made before the file system is compressed. In the event of a mishap during file system compression the file system could be restored from the backup.

Since the file system is destroyed during the compression process, it is not possible to compress the root file system. The `compress` command will reject the file system name `/` if it is entered.

Compressing any file system except `/usr` can be done through the `sysadm` command. An example of such a file compression is given below. Compressing the `/usr` file system is somewhat more complex a process, since the `sysadm` facilities reside in the `/usr` file system. A scheme for compressing `/usr` is given in the examples.

EXAMPLES

To compress a file system named *applic*, the following command would be issued:

```
sysadm tapemgmt
```

When the tape management facilities menu is displayed, select the `compress` facility and answer the questions posed by the shell procedure. This `compress` facility invokes the `/etc/cmpress` procedure.

To compress the `/usr` file system the UNIX system has to be in single user mode with the `/usr` file system mounted. The following sequence of commands will take the system from multiuser to single user mode, compress the `/usr` file system, then return the system to multiuser mode. Notice that any work going on in the system at the time that the system is changed to single user mode will be terminated, so this process should be done at a time when there are no other users logged in, and no background tasks are being done. It should be done only from the console from the root login.

First check to see how `/usr` is mounted.

```
mount
```

Make note of the `/dev/rSA/c?d?s?` information, as you will need it later.

Now take the system down to single user mode.

```
init 1
```

Lots of messages will now appear, and you will need to log back in as root.

Now mount the `/usr` file system. Use the `/dev/rSA/c?d?s?` information from the `mount` command above.

```
mount /dev/dsk/c1d0s2 /usr
```

Now compress the file system

```
/etc/cmpress
```

The procedure will pose a series of questions. As the compression process runs it will display a series of messages indicating its progress.

When compression is complete the following commands will unmount the `/usr` file system and return the system to multi user mode. Many messages will be displayed during this process.

```
umount /dev/dsk/c1d0s2  
init 2
```

SEE ALSO

`sysadm(1)` in the *User's Reference Manual*.

DIAGNOSTICS

The diagnostic messages are intended to be self explanatory.

WARNINGS

As mentioned above, since the compression of the file system entails its destruction and restoration it is strongly recommended that a backup copy of the file system be made before its compression is attempted.

NAME

crash – examine system images

SYNOPSIS

/etc/crash [*-d* *dumpfile*] [*-n* *namelist*] [*-w* *outputfile*]

DESCRIPTION

The *crash* command is used to examine the system memory image of a live or a crashed system by formatting and printing control structures, tables, and other information. Command line arguments to *crash* are *dumpfile*, *namelist*, and *outputfile*.

Dumpfile is the file containing the system memory image. The default *dumpfile* is */dev/mem*. The system image can also be */dev/ifdsk06*, if the first floppy of a system dump is taken with *sysdump(8)*; or it can be the pathname of a file produced by *ldsysdump(1M)*.

The text file *namelist* contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is */unix*. If a system image from another machine is to be examined, the corresponding text file must be copied from that machine.

When the *crash* command is invoked, a session is initiated. The output from a *crash* session is directed to *outputfile*. The default *outputfile* is the standard output.

Input during a *crash* session is of the form:

function [argument ...]

where *function* is one of the *crash* functions described in the "FUNCTIONS" section of this manual page, and *arguments* are qualifying data that indicate which items of the system image are to be printed.

The default for process-related items is the current process for a running system and the process that was running at the time of the crash for a crashed system. If the contents of a table are being dumped, the default is all active table entries.

The following function options are available to *crash* functions wherever they are semantically valid.

- e* Display every entry in a table.
- f* Display the full structure.
- p* Interpret all address arguments in the command line as *physical* addresses.
- s* process
Specify a process slot other than the default.
- w* file
Redirect the output of a function to *file*.

Note that if the *-p* option is used, all address and symbol arguments explicitly entered on the command line will be interpreted as physical addresses. If they are not physical addresses, results will be inconsistent.

The functions *mode*, *defproc*, and *redirect* correspond to the function options `-p`, `-s`, and `-w`. The *mode* function may be used to set the address translation mode to physical or virtual for all subsequently entered functions; *defproc* sets the value of the process slot argument for subsequent functions; and *redirect* redirects all subsequent output.

Output from *crash* functions may be piped to another program in the following way:

```
function [argument ... ]!shell_command
```

For example,

```
mount ! grep rw
```

will write all mount table entries with an *rw* flag to the standard output. The redirection option (`-w`) cannot be used with this feature.

Depending on the context of the function, numeric arguments will be assumed to be in a specific radix. Counts are assumed to be decimal. Addresses are always hexadecimal. Table address arguments larger than the size of the function table will be interpreted as hexadecimal addresses; those smaller will be assumed to be decimal slots in the table. Default bases on all arguments may be overridden. The C conventions for designating the bases of numbers are recognized. A number that is usually interpreted as decimal will be interpreted as hexadecimal if it is preceded by *0x* and as octal if it is preceded by *0*. Decimal override is designated by *0d*, and binary by *0b*.

Aliases for functions may be any uniquely identifiable initial substring of the function name. Traditional aliases of one letter, such as *p* for *proc*, remain valid.

Many functions accept different forms of entry for the same argument. Requests for table information will accept a table entry number, a physical address, a virtual address, a symbol, a range, or an expression. A range of slot numbers may be specified in the form *a-b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number; the operator may be `+`, `-`, `*`, `/`, `&`, or `|`. An operand which is a number should be preceded by a radix prefix if it is not a decimal number (*0* for octal, *0x* for hexadecimal, *0b* for binary). The expression must be enclosed in parentheses (`()`). Other functions will accept any of these argument forms that are meaningful.

Two abbreviated arguments to *crash* functions are used throughout. Both accept data entered in several forms. They may be expanded into the following:

```
table_entry = table entry|address|symbol|range|expression
```

```
start_addr = address|symbol|expression
```

FUNCTIONS

`? [-w file]` List available functions.

`!cmd` Escape to the shell to execute a command.

`adv [-e] [-w file] [[-p]table_entry...]`
Print the advertise table.

base [-w file] number ...

Print *number* in binary, octal, decimal, and hexadecimal. A number in a radix other than decimal should be preceded by a prefix that indicates its radix as follows: *0x*, hexadecimal; *0*, octal; and *0b*, binary.

buffer [-w file] [-format] bufferslot

or

buffer [-w file] [-format] [-p]start_addr

Alias: **b**.

Print the contents of a buffer in the designated format. The following format designations are recognized: **-b**, byte; **-c**, character; **-d**, decimal; **-x**, hexadecimal; **-o**, octal; **-r**, directory; and **-i**, inode. If no format is given, the previous format is used. The default format at the beginning of a *crash* session is hexadecimal.

bufhdr [-f] [-w file] [[-p]table_entry...]

Alias: **buf**.

Print system buffer headers.

The **-f** option produces different output depending on whether the buffer is local or remote (contains RFS data).

callout [-w file]

Alias: **c**.

Print the callout table.

dballoc [-w file] [class ...]

Print the dballoc table. If a class is entered, only data block allocation information for that class will be printed.

dbfree [-w file] [class ...]

Print free streams data block headers. If a class is entered, only data block headers for the class specified will be printed.

dblock [-e] [-w file] [-c class...]

or

dblock [-e] [-w file] [[-p] table_entry...]

Print allocated streams data block headers. If the class option (**-c**) is used, only data block headers for the class specified will be printed.

defproc [-w file] [-c]

or

defproc [-w file] [slot]

Set the value of the process slot argument. The process slot argument may be set to the current slot number (**-c**) or the slot number may be specified. If no argument is entered, the value of the previously set slot number is printed. At the start of a *crash* session, the process slot is set to the current process.

dis [-w file] [-a] start_addr [count]

Disassemble from the start address for *count* instructions. The default count is 1. The absolute option (**-a**) specifies a non-symbolic disassembly.

- ds** [-w file] virtual_address ...
 Print the data symbol whose address is closest to, but not greater than, the address entered.
- file** [-e] [-w file] [[-p]table_entry...]
 Alias: f.
 Print the file table.
- findaddr** [-w file] table slot
 Print the address of *slot* in *table*. Only tables available to the *size* function are available to *findaddr*.
- findslot** [-w file] virtual_address ...
 Print the table, entry slot number, and offset for the address entered. Only tables available to the *size* function are available to *findslot*.
- fs** [-w file] [[-p]table_entry...]
 Print the file system information table.
- gdp** [-e] [-f] [-w file] [[-p]table_entry...]
 Print the gift descriptor protocol table.
- help** [-w file] function ...
 Print a description of the named function, including syntax and aliases.
- inode** [-e] [-f] [-w file] [[-p]table_entry...]
 Alias: i.
 Print the inode table, including file system switch information.
- kfp** [-w file] [-s process] [-r]
 or
kfp [-w file] [-s process] [value]
 Print the frame pointer for the start of a kernel stack trace. The *kfp* value can be set using the value argument or the reset option (-r), which sets the *kfp* through the nvram. If no argument is entered, the current value of the *kfp* is printed.
- lck** [-e] [-w file] [[-p]table_entry...]
 Alias: l.
 Print record locking information. If the -e option is used or table address arguments are given, the record lock list is printed. If no argument is entered, information on locks relative to inodes is printed.
- linkblk** [-e] [-w file] [[-p]table_entry...]
 Print the linkblk table.
- major** [-w file] [entry ...]
 Print the MAJOR table.
- map** [-w file] mapname ...
 Print the map structure of the given mapname.
- mbfree** [-w file]
 Print free streams message block headers.

- mblock** [-e] [-w filename] [[-p]table_entry...]
Print allocated streams message block headers.
- mmu** [-w file]
Alias: **regs**.
Print memory management unit registers. These registers are not available on a running system.
- mode** [-w file] [mode]
Set address translation of arguments to virtual (**v**) or physical (**p**) mode. If no mode argument is given, the current mode is printed. At the start of a *crash* session, the mode is virtual.
- mount** [-e] [-w file] [[-p]table_entry...]
Alias: **m**.
Print the mount table.
- nm** [-w file] symbol ...
Print value and type for the given symbol.
- nvr** [-w file] type
Print nvr information of one of four types. *Type* may be **fwnvr**, **unxnvr**, **systate**, or **errlog**.
- od** [-p] [-w file] [-format] [-mode] [-s process] start_addr [count]
Alias: **rd**.
Print *count* values starting at the start address in one of the following formats: character (**-c**), decimal (**-d**), hexadecimal (**-x**), octal (**-o**), ascii (**-a**), or hexadecimal/character (**-h**), and one of the following modes: long (**-l**), short (**-t**), or byte (**-b**). The default mode for character and ascii formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format **-h** prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a *crash* session, the format is hexadecimal and the mode is long. If no count is entered, 1 is assumed.
- pcb** [-w file] [-u] [process]
or
pcb [-w file] [-k] [process]
or
pcb [-w file] [[-p]-i start_addr]
Print the process control block. If no arguments are given, the active pcb for the current process is printed. The user option (**-u**) prints the user pcb and the kernel option (**-k**) prints the kernel pcb associated with the process. The interrupt option (**-i**) prints the interrupt pcb located at the start address.
- pdt** [-e] [-w file] [-s process] section segment
or
pdt [-e] [-w file] [-s process] [-p] start_addr [count]
The page descriptor table of the designated memory *section* and

segment is printed. Alternatively, the page descriptor table starting at the start address for *count* entries is printed. If no count is entered, 1 is assumed.

pfdat [-e] [-w file] [[-p]table_entry...]

Print the pfdata table.

proc [-e] [-f] [-w file] [[-p] table_entry ... #procid ...]

or

proc [-f] [-w file] [-r]

Alias: p.

Print the process table. Process table information may be specified in two ways. First, any mixture of table entries and process ids may be entered. Each process id must be preceded by a #. Alternatively, process table information for runnable processes may be specified with the runnable option (-r).

qrun [-w file]

Print the list of scheduled streams queues.

queue [-e] [-w file] [[-p]table_entry...]

Print streams queues.

quit Alias: q.

Terminate the *crash* session.

rcvd [-e] [-f] [-w file] [[-p]table_entry...]

Print the receive descriptor table.

redirect [-w file] [-c]

or

redirect [-w file] [file]

Used with a file name, redirects output of a *crash* session to the named file. If no argument is given, the file name to which output is being redirected is printed. Alternatively, the close option (-c) closes the previously set file and redirects output to the standard output.

region [-e] [-f] [-w file] [[-p]table_entry...]

Print the region table.

sdt [-e] [-w file] [-s process] section

or

sdt [-e] [-w file] [-s process] [-p] start_addr[count]

The segment descriptor table for the named memory section is printed. Alternatively, the segment descriptor table starting at start address for *count* entries is printed. If no count is given, a count of 1 is assumed.

search [-p] [-w file] [-m mask] [-s process] pattern start_addr length

Print the words in memory that match *pattern*, beginning at the start address for *length* words. The mask is anded (&) with each memory word and the result compared against the pattern. The mask defaults to 0xffffffff.

size [-w file] [-x] [structure_name ...]
Print the size of the designated structure. The (-x) option prints the size in hexadecimal. If no argument is given, a list of the structure names for which sizes are available is printed.

sndd [-e] [-f] [-w file] [[-p]table_entry...]
Print the send descriptor table.

srams [-w file]
Print the sram values.

srmount [-e] [-w file] [[-p]table_entry...]
Print the server mount table.

stack [-w file] [-u] [process]
or
stack [-w file] [-k] [process]
or
stack [-w file] [[-p]-i start_addr]
Alias: s.
Dump stack. The (-u) option prints the user stack. The (-k) option prints the kernel stack. The (-i) option prints the interrupt stack starting at the start address. If no arguments are entered, the kernel stack for the current process is printed. The interrupt stack and the stack for the current process are not available on a running system.

stat [-w file]
Print system statistics.

stream [-e] [-f] [-w file] [[-p]table_entry...]
Print the streams table.

strstat [-w file]
Print streams statistics.

trace [-w file] [-r] [process]
or
trace [-w file] [[-p]-i start_addr]
Alias: t.
Print stack trace. The kfp value is used with the -r option. The interrupt option prints a trace of the interrupt stack beginning at the start address. The interrupt stack trace and the stack trace for the current process are not available on a running system.

ts [-w file] virtual_address ...
Print closest text symbol to the designated address.

tty [-e] [-f] [-w file] [-t type][[-p]table_entry...]
or
tty [-e] [-f] [-w file] [[-p]start_addr]
Valid types: **pp**, **iu**.
Print the tty table. If no arguments are given, the tty table for both tty

types is printed. If the `-t` option is used, the table for the single tty type specified is printed. If no argument follows the type option, all entries in the table are printed. A single tty entry may be specified from the start address.

user [`-f`] [`-w file`] [`process`]
Alias: `u`.
Print the ublock for the designated process.

var [`-w file`]
Alias: `v`.
Print the tunable system parameters.

vtop [`-w file`] [`-s process`] `start_addr...`
Print the physical address translation of the virtual start address.

FILES

<code>/dev/mem</code>	system image of currently running system
<code>/dev/ufdisk06</code>	used to access system image on floppy diskette

SEE ALSO

`ldsysdump(1M)`, `sysdump(8)`.

NAME

`cron` – clock daemon

SYNOPSIS

`/etc/cron`

DESCRIPTION

`cron` executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in `crontab` files in the directory `/usr/spool/cron/crontabs`. Users can submit their own `crontab` file via the `crontab(1)` command. Commands which are to be executed only once may be submitted via the `at(1)` command.

`cron` only examines `crontab` files and `at` command files during process initialization and when a file changes via `crontab` or `at`. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

Since `cron` never exits, it should be executed only once. This is done routinely through `/etc/rc2.d/S75cron` at system boot time. `/usr/lib/cron/FIFO` is used as a lock file to prevent the execution of more than one `cron`.

FILES

<code>/usr/lib/cron</code>	main <code>cron</code> directory
<code>/usr/lib/cron/FIFO</code>	used as a lock file
<code>/usr/lib/cron/log</code>	accounting information
<code>/usr/spool/cron</code>	spool area

SEE ALSO

`at(1)`, `crontab(1)`, `sh(1)` in the *User's Reference Manual*.

DIAGNOSTICS

A history of all actions taken by `cron` are recorded in `/usr/lib/cron/log`.



NAME

ctccpio – copy file archives in and out to cartridge tape

SYNOPSIS

/etc/ctccpio -o [avVK] -T cartridge_tape_device

/etc/ctccpio -i [dmrtuvVf] -T cartridge_tape_device [patterns]

DESCRIPTION

ctccpio -o (copy out) reads the standard input to obtain a list of path names and copies those files together with path name and status information to the cartridge tape device specified on the command line (**-T cartridge_tape_device**). Always use the **-K** option with this command to verify that the data was copied.

ctccpio -i (copy in) extracts files from the cartridge tape device specified on the command line (**-T cartridge_tape_device**), which is assumed to be the product of a previous **ctccpio -o**. Only files with names that match *patterns* are selected. *Patterns* are given in the name-generating notation of *sh*(1). In *patterns*, meta-characters *?*, ***, and *[...]* match the slash */* character. Multiple *patterns* may be specified and if no *patterns* are specified, the default for *patterns* is *** (i.e., select all files). The extracted files are conditionally created and copied into the current directory tree based upon the options described below. The permissions of the files will be those of the previous **ctccpio -o**. The owner and group of the files will be that of the current user unless the user is super-user, which causes *ctccpio* to retain the owner and group of the files of the previous **ctccpio -o**.

ctccpio is a modified version of *cpio* that uses the streaming feature of the cartridge tape controller. *ctccpio* will provide better performance than *cpio* when using the cartridge tape, although the amount of streaming that is achieved will depend upon the size of the files to be copied, file system fragmentation, etc. Headers are always written in ASCII character format.

The meanings of the available options are:

- a** Reset access times of input files after they have been copied.
- d** *Directories* are to be created as needed.
- r** Interactively *rename* files. If the user types a null line, the file is skipped.
- t** Print a *table of contents* of the input. No files are created.
- u** Copy *unconditionally* (normally, an older file will not replace a newer file with the same name).
- v** *Verbose*: causes a list of file names to be printed. When used with the *t* option, the table of contents looks like the output of an *ls -l* command [see *ls*(1)].
- V** Print out a dot for each file copied.
- m** Retain previous file modification time. This option is ineffective on directories that are being copied.
- f** Copy in all files except those in *patterns*.
- K** Perform verify pass on output to tape. This option will significantly decrease the performance of *ctccpio*.

EXAMPLES

The first example below copies the contents of a directory to the tape device `/dev/rSA/ctape1`; the second reads the tape and writes the files copied in the first example:

```
cd olddir
find -print | ctccpio -oKT /dev/rSA/ctape1
cd newdir
ctccpio -iT /dev/rSA/ctape1
```

SEE ALSO

`cpio(4)`,
`ar(1)`, `find(1)`, `ls(1)` in the *User's Reference Manual*.

WARNING

Only partition 3 (`/dev/rSA/ctape1`) is recommended for streaming. Use of other partitions that begin at the tape's logical block 0 will corrupt tape format data and require re-formatting the tape.

BUGS

Path names are restricted to 128 characters. If there are too many unique linked files, the program runs out of memory to keep track of them and, thereafter, linking information is lost. Only the super-user can copy special files.

NAME

`ctcfmt` – format cartridge tape

SYNOPSIS

```
/etc/ctcmft [-v] [-p passct] -t rawdevice
```

DESCRIPTION

`ctcfmt` is used to format or reformat a tape cartridge on the 3B2 Computer. The parameter *rawdevice* specifies the name of the device, as a raw device name, of the drive to be used for the formatting operation.

The option `-p` indicates that the maximum number of tape passes to be allowed for the tape is to be set to *passct*. This number is used to warn the user that the tape is approaching the end of its lifetime and should be replaced to avoid the loss of data. The default value of *passct* is 4000.

The option `-v` indicates that the format is to be verified after being done. Use of this option will prevent a detectably bad tape from being used, since verification will read the format information from every block on the tape and detect unreadable blocks. Because it makes two passes over the medium and checks every block on it, the time needed to format a medium with this option is noticeably longer than without this option. The decision of whether or not to verify should be based on the importance of the data to be placed on the medium, and on the degree to which the medium currently being used by the user exhibits defects. If the `-v` option is not used defects are not mapped.

EXAMPLES

The following command formats a floppy tape with 4000 as the maximum pass count and does not verify the formatting operation:

```
ctcfmt -t /dev/rSA/ctape1
```

The following command does a format with verification and sets the pass count to 3500:

```
ctcfmt -v -p 3500 -t /dev/rSA/ctape2
```

DIAGNOSTICS

The diagnostic messages are intended to be self explanatory.

WARNINGS

Once a tape has been formatted, any data previously on the medium is lost.

Always use the verify option when formatting cartridge tapes. The verify pass puts a defect map on the cartridge tape. This effectively eliminates bad blocks on a cartridge tape by mapping around them.



NAME

ctcinfo – display information about cartridge tape

SYNOPSIS

/etc/ctcinfo [options] rawdevice

DESCRIPTION

ctcinfo displays certain subdevice information for the Cartridge Tape Controller (CTC). It also can be used to reset the usage count for a tape drive after it has been cleaned. The drive to be used is specified as the raw device *rawdevice*.

The display options and the information printed by the program are the following:

- v Volume Table of Contents (vtoc)
- d Device Type
- t Tape Pass Count
- m Maximum Tape Pass Count
- u Tape Drive Usage Count
- c Number of Cylinders
- x Number of Tracks per Cylinder
- s Number of Sectors per Track
- b Number of Bytes per Sector
- a Total Number of Bytes on Tape
- B Total Number of Blocks on Tape

As a special case, the option –r resets the tape drive usage count. It should only be used to inform the system that the tape drive has been cleaned so that the system will cease issuing warning messages about the danger of running with a dirty tape drive. Using this option to turn off the warnings without cleaning the tape drive is extremely hazardous to the health of the data stored on all the tapes passing through the drive and simply should not be done.

EXAMPLES

The following example asks for a display of the Volume Table of Contents for a tape:

```
/etc/ctcinfo -v /dev/rSA/ctape1
```

The following example illustrates a request to display several items of information about a tape:

```
/etc/ctcinfo -d -t -c -a /dev/rSA/ctape1
```

The following example shows how to reset the usage count after cleaning the tape drive, then see how much time is left until the next cleaning:

```
/etc/ctcinfo -u -r /dev/rSA/ctape1
```

Note that the usage count is reset before any of the other options are acted on.

DIAGNOSTICS

The diagnostic messages are intended to be self explanatory.

WARNINGS

Again, it should be reiterated that the **-r** option should never be used unless the tape drive has just been cleaned. You can fool the system with this command, but you can also thusly destroy irreplaceable data.

NAME

`dcopy` – copy file systems for optimal access time

SYNOPSIS

`/etc/dcopy [-sX] [-an] [-d] [-v] [-ffsize[:isize]] inputfs outputfs`

DESCRIPTION

`dcopy` copies file system *inputfs* to *outputfs*. *Inputfs* is the device file for the existing file system; *outputfs* is the device file to hold the reorganized result. For the most effective optimization *inputfs* should be the raw device and *outputfs* should be the block device. Both *inputfs* and *outputfs* should be unmounted file systems (in the case of the root file system, the copy must be to a new pack).

With no options, `dcopy` copies files from *inputfs* compressing directories by removing vacant entries, and spacing consecutive blocks in a file by the optimal rotational gap. The possible options are

- `-sX` supply device information for creating an optimal organization of blocks in a file. The forms of *X* are the same as the `-s` option of `fsck(1M)`.
- `-an` place the files not accessed in *n* days after the free blocks of the destination file system (default for *n* is 7). If no *n* is specified then no movement occurs.
- `-d` leave order of directory entries as is (default is to move sub-directories to the beginning of directories).
- `-v` currently reports how many files were processed, and how big the source and destination freelists are.
- `-ffsize[:isize]` specify the *outputfs* file system and inode list sizes (in blocks). If the option (or *:isize*) is not given, the values from the *inputfs* are used.

`dcopy` catches interrupts and quits, and reports on its progress. To terminate `dcopy` send a quit signal, followed by an interrupt or quit.

SEE ALSO

`fsck(1M)`, `mkfs(1M)`.
`ps(1)` in the *User's Reference Manual*.



NAME

dd – convert and copy a file

SYNOPSIS

dd [option=value] ...

DESCRIPTION

dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block size may be specified to take advantage of raw physical I/O.

<i>option</i>	<i>values</i>
if=file	input file name; standard input is default
of=file	output file name; standard output is default
ibs=n	input block size <i>n</i> bytes (default 512)
obs=n	output block size (default 512)
bs=n	set both input and output block size, superseding <i>ibs</i> and <i>obs</i> ; also, if no conversion is specified, it is particularly efficient since no in-core copy need be done
cbs=n	conversion buffer size
skip=n	skip <i>n</i> input blocks before starting copy
seek=n	seek <i>n</i> blocks from beginning of output file before copying
count=n	copy only <i>n</i> input blocks
conv=ascii	convert EBCDIC to ASCII
ebcdic	convert ASCII to EBCDIC
ibm	slightly different map of ASCII to EBCDIC
lcase	map alphabetic to lower case
ucase	map alphabetic to upper case
swab	swap every pair of bytes
noerror	do not stop processing on an error
sync	pad every input block to <i>ibs</i>
... , ...	several comma-separated conversions

Where sizes are specified, a number of bytes is expected. A number may end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by **x** to indicate multiplication.

cbs is used only if *conv=ascii* or *conv=ebcdic* is specified. In the former case, *cbs* characters are placed into the conversion buffer (converted to ASCII). Trailing blanks are trimmed and a new-line added before sending the line to the output. In the latter case, ASCII characters are read into the conversion buffer (converted to EBCDIC). Blanks are added to make up an output block of size *cbs*.

After completion, *dd* reports the number of whole and partial input and output blocks.

DIAGNOSTICS

f+p blocks in(out) numbers of full and partial blocks read(written)



NAME

`devinfo` – print device specific information

SYNOPSIS

`/usr/sbin/devinfo -i | -p special`

DESCRIPTION

The *devinfo* command is used to print device specific information about disk devices on standard out.

The options have the following effect:

-i option will print the following device information:

Device name	Software version
Drive identification number	Device blocks per cylinder
Device bytes per block	Number of device partitions with a block size greater than zero

-p will print the following device partition information:

Device name	Device major and minor numbers
Partition start block	Number of blocks allocated to the partition
Partition flag	Partition tag

The command is used by various other commands to obtain device specific information for the making of file systems and determining partition information.

SEE ALSO

`prvtoc(1M)`.



NAME

devnm – device name

SYNOPSIS

/etc/devnm [names]

DESCRIPTION

devnm identifies the special file associated with the mounted file system where the argument *name* resides.

This command is most commonly used by */etc/brc* (see *brc(1M)*) to construct a mount table entry for the **root** device.

EXAMPLE

The command:

```
/etc/devnm /usr
```

produces

```
/dev/dsk/c1d0s2 usr
```

if **/usr** is mounted on **/dev/dsk/c1d0s2**.

FILES

/dev/dsk/*

/etc/mnttab

SEE ALSO

brc(1M).



NAME

df -- report number of free disk blocks and i-nodes

SYNOPSIS

df [-lt] [-f] [*file-system* | *directory* | *mounted-resource*]

DESCRIPTION

The **df** command prints out the number of free blocks and free i-nodes in mounted file systems, directories, or mounted resources by examining the counts kept in the super-blocks.

file-system may be specified either by device name (e.g., */dev/dsk/c1d0s2*) or by mount point directory name (e.g., */usr*).

directory can be a directory name. The report presents information for the device that contains the directory.

mounted-resource can be a remote resource name. The report presents information for the remote device that contains the resource.

If no arguments are used, the free space on all locally and remotely mounted file systems is printed.

The **df** command uses the following options:

- l only reports on local file systems.
- t causes the figures for total allocated blocks and i-nodes to be reported as well as the free blocks and i-nodes.
- f an actual count of the blocks in the free list is made, rather than taking the figure from the super-block (free i-nodes are not reported). This option will not print any information about mounted remote resources.

NOTE

If multiple remote resources are listed that reside on the same file system on a remote machine, each listing after the first one will be marked with an asterisk.

FILES

*/dev/dsk/**
/etc/mnttab

SEE ALSO

mount(1M), *fs(4)*, *mnttab(4)* in the *System Administrator's Reference Manual*.



NAME

`disks` – adds `/dev/entries` for hard disks in the Equipped Device Table

SYNOPSIS

`/etc/disks`

DESCRIPTION

`disks` will search the Equipped Device Table (EDT) to see which hard disks are equipped. For each equipped hard disk, the following steps are performed:

1. The `/dev/dsk` and `/dev/rdisk` directories are checked for an entry with the name `q[slot]d?s6`. Where `[slot]` is the slot the disk controller board is plugged into which is 0 for the disks controlled by the integral disk controller on the system board. The `?` is the number of the disk attached to the controller. The system board disk controller is capable of controlling two disks: 0 and 1.
2. If either entry is not found, `disks` creates `/dev/dsk` and `/dev/rdisk` entries for the disk. The `/dev/SA` and `/dev/rSA` entries are created and linked to the `q[slot]d?s6` entry in `/dev/dsk` and `/dev/rdisk` respectively. The `/dev/SA` and `/dev/rSA` entries are named `diskx` where `x` is the lowest unused number for disk entries. A message is printed indicating that `/dev` files have been created.

`disks` is called each time the system is booted. It must also be called after `"sysadm rmdisk"` to restore the `/dev` entries so the disk can be repartitioned.

FILES

`/dev/dsk/*` entries for the hard disk for general use
`/dev/rdisk/*`
`/dev/SA/*` entries for the hard disk for use by System Administration
`/dev/rSA/*`

SEE ALSO

`sysadm(1)` in the *User's Reference Manual*.



NAME

dname – Print Remote File Sharing domain and network names

SYNOPSIS

dname [-D *domain*] [-N *netspec*] [-dna]

DESCRIPTION

dname prints or defines a host's Remote File Sharing domain name or the network used by Remote File Sharing as transport provider. When used with **d**, **n**, or **a** options, **dname** can be run by any user to print the domain name, network name or both, respectively. Only a user with root permission can use the **-D domain** option to set the domain name for the host or **-N netspec** to set the network specification used for Remote File Sharing. (The value of *netspec* is the network device name, relative to the */dev* directory. For example, the STARLAN NETWORK uses **starlan**.)

domain must consist of no more than 14 characters, consisting of any combination of letters (upper and lower case), digits, hyphens (-), and underscores (_)

When **dname** is used to change a domain name, the host's password is removed. The administrator will be prompted for a new password the next time Remote File Sharing is started [*rfstart*(1M)].

If **dname** is used with no options, it will default to **dname -d**.

ERRORS

You cannot use the **-N** or **-D** options while Remote File Sharing is running.

SEE ALSO

rfstart(1M).



NAME

`drvinstall` – install/uninstall a driver

SYNOPSIS

```
/etc/drvinstall [ -m master ] [ -d object ] [ -s system ]
[ -o directory ] [ -c minor ] -v version -ufbnx
```

DESCRIPTION

The `drvinstall` command accepts an *object* file, *master* file and *system* file as inputs, and creates the corresponding specially formatted file for use in the self-configuring boot process. In addition, the *master* and *system* files may be modified.

-m master

argument specifies the path name of the *master* file to be used. If this flag is omitted, the `/etc/master.d` directory is used.

-d object

argument specifies the path name of the input *object* file to be used. If this flag is omitted, the `/boot` directory is used.

One or both of the **-m** or **-d** options must be specified.

-s system

argument specifies the path name of the *system* file to be used. If this flag is omitted, the `/etc/system` file is used.

-o directory

argument specifies the path name of the output bootable file. If this flag is omitted, the `/boot` directory is used.

-v version

argument specifies the *version* number of `drvinstall` command compatible with the *master* file being used. The **-v** option is required on the command line and currently supports "1.0".

If the **-u** option is not present, the driver will be installed. If the driver to be installed is a software driver, `drvinstall` will:

- Assign a major number to that driver if there is an "-" entry in the major number field of the associated *master* file entry. The `drvinstall` command expects any unused field of the *master* file to be filled with a "-".

The major numbers available for software drivers on AT&T 3B2 computers are 30–71 inclusive. The remaining major numbers are reserved for hardware devices, used by integral drivers, or reserved for SCSI devices. The `drvinstall` command determines the available major numbers by scanning all existing *master* files for major numbers and then assigning the first unused number in the above range. This value replaces the corresponding "-" value in the major number field of the *master* file.

- Print the major number found or assigned in the *master* file.

If the driver to be installed is not a hardware driver (it is, e.g., a software driver or a loadable type of module), `drvinstall` will:

- Insert an INCLUDE statement for the driver in the *system* file.

- If the `-c` option is present, insert "*minor*" at the end of the `INCLUDE` statement. *Minor* is optional in an `INCLUDE` and specifies the quantity (default of 1) of minor devices to be controlled by the driver. If the driver is not a software driver, `-c` is ignored.

For any driver installed, *drvinstall* calls *mkboot*(1M) to produce a bootable *object* file. The resultant output file is placed into the directory determined by the `-o` argument. The `-b` option inhibits generation of the *object* file. This option is ignored for `uninstall`.

The `-u` option will uninstall a driver. A driver dependency check is made and if a dependency is found, a warning message is issued and the command is aborted. If no dependency is found, then *drvinstall* will:

- Remove the bootable *object* file.
- Replace the major number with a "-" in the *master* file if the driver is a software driver.
- Delete the `INCLUDE` statement from the *system* file if the driver is not a hardware driver.
- Print the major number if the driver is a software driver.

The `-f` option, when used with the `-u` option, disables the dependency check. This results in the driver being uninstalled regardless of dependencies.

The following options apply to both installing and uninstalling:

- `-n` Inhibits any edit of the *system* file.
- `-x` Enables debugging output.

SEE ALSO

mkboot(1M).
master(4), *system*(4) in the *Programmer's Reference Manual*.

DIAGNOSTICS

The major number assigned or found for a software driver is printed on `stdout`. A zero is returned for success and a non-zero is returned for failures.

NAME

`du` – summarize disk usage

SYNOPSIS

`du [-sar] [names]`

DESCRIPTION

du reports the number of blocks contained in all files and (recursively) directories within each directory and file specified by the *names* argument. The block count includes the indirect blocks of the file. If *names* is missing, the current directory is used.

The optional arguments are as follows:

- `-s` causes only the grand total (for each of the specified *names*) to be given.
- `-a` causes an output line to be generated for each file.

If neither `-s` or `-a` is specified, an output line is generated for each directory only.

- `-r` will cause *du* to generate messages about directories that cannot be read, files that cannot be opened, etc., rather than being silent (the default).

A file with two or more links is only counted once.

BUGS

If the `-a` option is not used, non-directories given as arguments are not listed. If there are links between files in different directories where the directories are on separate branches of the file system hierarchy, *du* will count the excess files more than once.

Files with holes in them will get an incorrect block count. (See Chapter 5, File System Administration, in the *System Administrator's Guide*)



NAME

*edit*sa – add/delete entry from software application file

SYNOPSIS

*/etc/edit*sa –i slot *HWNAME SWNAME*

*/etc/edit*sa –r *SWNAME*

*/etc/edit*sa –l

DESCRIPTION

The *edit*sa command is used to add, delete or list entries in the software application file, */dgn/.edt_swapp*. It is primarily used in application installation scripts to modify the software application file. This file allows a software driver, *SWNAME*, to be associated with a specific board in a specific 3B2 Computer expansion slot. The file, */dgn/.edt_swapp*, is a data base which is read by *filledt(1M)* to update the *name* field for designated Equipped Device Table (EDT) entries. *edit*sa performs various checks to ensure the request will result in a valid system configuration.

The valid uses of *edit*sa are as shown above and will be returned if not properly entered. The argument definitions are as follows:

SWNAME

Specifies the name of the driver which is being dynamically assigned to a hardware device or removed as a valid entry in the software application file.

HWNAME

Specifies the name of the hardware device (e.g. PORTS, ISC, NI) for which the EDT entry is being renamed.

slot This argument specifies the slot entry in the EDT for which the board, *HWNAME*, is to be replaced by the argument specified as *SWNAME*

–i This option specifies that the corresponding entry should be added to the software application file.

–r All entries which match the *SWNAME* specified in the command line will be removed from the software application file.

–l This option will print a formatted display of the software application file.

FILES

*/etc/edit*sa

/etc/filledt

/dgn/.edt_swapp

DIAGNOSTICS

All errors from *edit*sa are fatal and return an error code of 1 with the exception of a warning message for the case when the *HWNAME* specified does not match the EDT entry for the slot requested on the command line. This warning returns an error code of 2.



NAME

`edittbl` – edit `edt_data` file

SYNOPSIS

`/etc/edittbl -d[-s [-g] [-i] [-l] [-r] [-t] [-B bus_type] [file]`

DESCRIPTION

`edittbl` is a user-level utility that permits changes to `edt_data` files. These files are used by the firmware program `filledt(8)` during the Equipped Device Table (EDT) construction.

`edittbl` prints the option list when no arguments are specified. The arguments are:

- `-d` Selects the device look-up table for the system bus. This option cannot be used with the `-B` option.
- `-s` Selects the sub-device look-up table for the system bus. This option cannot be used with the `-B` option.
- `-g` Generates the default entries for the selected look-up table(s). This overwrites any entries that are currently in the selected look-up table(s). For the device table, these base entries are SBD, and PORTS. For the subdevice table, they are NULL, FD5, HD10, HD30, HD72, HD72A, HD72B, HD72C, HD43, and HD72D.
- `-i` Specifies that new entries are to be added to the selected table. The ID codes for table entries and the input are compared; only new codes are installed. The formats for entries are described below. An EOF or “.” ends the data input.
- `-l` Specifies that the selected table(s) are listed.
- `-r` Specifies that entries are to be removed from either table. The ID codes of the table are compared to the input; entries with codes that match the input are removed. The format is identical to that for the `-i` option and is listed below. An EOF or “.” ends the data input.
- `-t` Suppresses the program headings and user prompts; warnings and errors are not affected. This option is primarily useful in installation and removal scripts.
- `-B` Specifies that the `edt_data` file for `bus_type` is to be edited. Currently the only bus type supported is `scsi`. If this option is not specified, the system bus `edt_data` file will be edited.

file The user may specify a target path name for the utilities. If none is specified, `./edt_data` is the default. For the `scsi` bus type, *file* is where the target controller information is to be placed. If *file* is not supplied, the default name of `/edt/SCSI/edt_data` is used.

INPUT FORMAT

Data for installation/removal are entered as hex format numbers or character strings, one line for each table entry. The data fields must be supplied in the sequence described.

Devices

ID_code	is a number between 0x0 and 0xffff that a device uses to identify itself. For extended devices (such as SCSI), add 0x10000 to the ID_code. ID codes are administered by AT&T.
name	field name for a device. Device names are administered by AT&T. This string is also the file name that DGMON loads to diagnose a device.
rq_size	is a number between 0x0 and 0xff for the count of entries in a device's job request queue.
cq_size	is a number between 0x0 and 0xff for the count of entries in a device's job completion queue.
boot_device	determines whether a device may be used to boot programs. A "1" means that it is bootable; a "0" means that it is not.
word_size	shows the word size of a device I/O bus. A "1" is used for devices with a 16 bit bus word; a "0" is used for devices with an 8 bit bus word.
brd_size	specifies the I/O connector slots that a device requires. A "1" indicates that two slots are needed, while a "0" means that one is required.
smart_board	determines whether a device is intelligent, i.e., requires downloaded code for normal operation or supports subdevices. A "1" indicates an intelligent device; a "0" specifies a "dumb" device.
cons_cap	shows whether a device can support the system console terminal. A "1" is used for devices that can, a "0" for those that cannot.
cons_file	shows whether a device needs downloaded code to support the console interface when cons_cap has a value of "1". A "0" in this field means that the device can support a system console terminal with PROM-based code. A "1" in this field means downloaded code is needed. cons_file must have the value of "0" when cons_cap is "0".

Subdevices

ID_code	a number between 0x0 and 0xffff for the code that identifies a subdevice. Subdevice ID codes are administered by AT&T.
subdev_name	is a string (maximum of 9 characters) for a subdevice name. Subdevice names are administered by AT&T.
dev_name	is a string (maximum of 9 characters) for the device name with which the subdevice is associated.

EXAMPLES

Generate and list the base entries for both the device and subdevice tables, saving the results in `./edt_data`.

```
edittbl -g -l -s -d
```

Install subdevice entries with new ID codes from the file `subdev.in` into the existing file `./edt_data`.

```
edittbl -i -s < subdev.in
```

List the device table entries found in *file*

```
edittbl -l -d /dgn/edt_data
```

FILES

```
/dgn/edt_data  
/edt/SCSI/edt_data  
/etc/scsi/edittbl
```



NAME

errdump – print error log

SYNOPSIS*/etc/errdump***DESCRIPTION**

This command displays on the system console the error log contained in the system's nonvolatile ram. The display contains the previous saved system state, the last 5 panic messages and their time of occurrence, and an indication of the log's sanity.

DIAGNOSTICS

The phrase "not superuser" is displayed, if the command is invoked by other than the super-user. Super-user is defined as anyone logged in under the root directory from the console port.

EXAMPLE

The following is an example of the printout in response to the *errdump* command.

```
#
#
#
# errdump
nvram status:   sane

csr:    0x0648  (floppy)  (unassigned)  (clock)  (uart)

psw:    rsvd  CSH_F_D  QIE  CSH_D  OE  NZVC  TE  IPL  CM  PM  R  I  ISC  TM  FT
         0      1    0      1    0    0    0    f    0    0  1  0    5    0    3

r3:     0x00049001
r4:     0x00000081
r5:     0x00000000
r6:     0x40091348
r7:     0x0001a13f
r8:     0x4008edd8
oap:    0x400816d8
opc:    0x40083bc
osp:    0x40081700
ofp:    0x40081700
isp:    0x40080008
pcbp:   0x40041a40

fltar:  0xc0021140
fltcr:  reqacc  xlevel  ftype
        0xa     0x0     0x0
```

```
      srama      sramb
[0] 0x02034800 0x0000011f
[1] 0x02035100 0x00000030
[2] 0x02035860 0x00000074
[3] 0x02035c00 0x00000015
    Panic log

[0] Thu Sep 20 09:51:36 1984
    KERNEL DATA ALIGNMENT ERROR

[1] Thu Sep 20 09:51:37 1984
    KERNEL DATA ALIGNMENT ERROR

[2] Thu Sep 20 09:51:40 1984
    KERNEL DATA ALIGNMENT ERROR

[3] Thu Sep 20 09:52:21 1984
    KERNEL DATA ALIGNMENT ERROR

[4] Fri Sep 21 05:50:10 1984
    SYSTEM PARITY ERROR INTERRUPT
```

SEE ALSO

Appendix C: Error Messages in the *System Administrator's Guide*.

NAME

`ff` – list file names and statistics for a file system

SYNOPSIS

`/etc/ff` [*options*] *special*

DESCRIPTION

`ff` reads the *i*-list and directories of the *special* file, assuming it is a file system. *i*-node data is saved for files which match the selection criteria. Output consists of the path name for each saved *i*-node, plus other file information requested using the print *options* below. Output fields are positional. The output is produced in *i*-node order; fields are separated by tabs. The default line produced by `ff` is:

path-name i-number

With all *options* enabled, output fields would be:

path-name i-number size uid

The argument *n* in the *option* descriptions that follow is used as a decimal integer (optionally signed), where $+n$ means more than *n*, $-n$ means less than *n*, and *n* means exactly *n*. A day is defined as a 24 hour period.

- `-I` Do not print the *i*-node number after each path name.
- `-l` Generate a supplementary list of all path names for multiply-linked files.
- `-p prefix` The specified *prefix* will be added to each generated path name. The default is `.` (dot).
- `-s` Print the file size, in bytes, after each path name.
- `-u` Print the owner's login name after each path name.
- `-a n` Select if the *i*-node has been accessed in *n* days.
- `-m n` Select if the *i*-node has been modified in *n* days.
- `-c n` Select if the *i*-node has been changed in *n* days.
- `-n file` Select if the *i*-node has been modified more recently than the argument *file*.
- `-i i-node-list` Generate names for only those *i*-nodes specified in *i-node-list*.

SEE ALSO

`ncheck(1M)`.
`find(1)` in the *User's Reference Manual*.

BUGS

If the `-l` option is not specified, only a single path name out of all possible ones is generated for a multiply-linked *i*-node. If `-l` is specified, all possible names for every linked file on the file system are included in the output. However, no selection criteria apply to the names generated.



NAME

`finc` – fast incremental backup

SYNOPSIS

`/etc/finc [selection-criteria] file-system raw-tape`

DESCRIPTION

`finc` selectively copies the input *file-system* to the output *raw-tape*. The cautious will want to mount the input *file-system* read-only to insure an accurate backup, although acceptable results can be obtained in read-write mode. The tape must be previously labelled by `labelit`. The selection is controlled by the *selection-criteria*, accepting only those inodes/files for whom the conditions are true.

It is recommended that production of a `finc` tape be preceded by the `ff` command, and the output of `ff` be saved as an index of the tape's contents. Files on a `finc` tape may be recovered with the `frec` command.

The argument *n* in the *selection-criteria* which follow is used as a decimal integer (optionally signed), where `+n` means more than *n*, `-n` means less than *n*, and *n* means exactly *n*. A day is defined as a 24 hours.

- `-a n` True if the file has been accessed in *n* days.
- `-m n` True if the file has been modified in *n* days.
- `-c n` True if the i-node has been changed in *n* days.
- `-n file` True for any file which has been modified more recently than the argument *file*.

EXAMPLES

To write a tape consisting of all files from *file-system* `/usr` modified in the last 48 hours:

```
finc -m -2 /dev/rdsk/c1d0s2 /dev/rSA/ctape1
```

SEE ALSO

`ff(1M)`, `frec(1M)`, `labelit(1M)`.
`cpio(1)` in the *User's Reference Manual*.



NAME

fltboot – set NVRAM parameters for floating boot

SYNOPSIS

/etc/fltboot

DESCRIPTION

The **fltboot** command provides the supporting routines for setting or modifying the floating boot parameters of the 3B2 Computer. The floating boot feature allows the user to change the boot parameters which are the default boot file and the device on which the boot file is resident. It is recommended that direct execution of this command be avoided. The recommended method for changing floating boot parameters is by invoking the command **sysam autold**. This method provides a safe, interactive means to change parameter(s) by prompting the user for the desired changes.

SEE ALSO

sysadm(1) in the *User's Reference Manual*.



NAME

fmtflop – physically format diskettes

SYNOPSIS

/etc/mtflop [*-v*] *special_file*

DESCRIPTION

fmtflop physically formats the media inserted in the diskette drive. The *-v* option verifies that the diskette is correctly formatted. The *special_file* is the path name of the diskette drive (e.g., */dev/rdisk/c0d0s6*).

fmtflop formats DOUBLE SIDED media with 512 byte sectors, 9 sectors per track, and 80 tracks. Before executing *fmtflop*, the diskette must be placed in the drive and the latch closed.

SEE ALSO

if(7).

DIAGNOSTICS

An error message is returned if the format or verify fails. If this occurs, remove the diskette and reinsert it to make sure it is properly seated, then try entering the command a second time. If the command fails again (especially on the same area of the disk) the diskette is probably bad and must be discarded.

The error message "process lock failed" is returned if the command is attempted by someone other than the super-user.



NAME

`fmthard` – populate VTOC on hard disks

SYNOPSIS

```
/etc/fmthard [ -c core_disk_type ] [ -d data ] [ -i ] [ -m ] [ -s datafile ]
[ -n volume_name ] /dev/rdisk/c?[t?]d?s?
```

DESCRIPTION

The `fmthard` command creates (or updates) the VTOC (Volume Table of Contents) on "hard" disks. The `/dev/rdisk/c?[t?]d?s?` file must be the character special file of the device where the new VTOC is to be installed.

OPTIONS

The following options apply to `fmthard`:

-d *data*

The *data* argument of this option is a string representing the information for a particular partition in the current VTOC. The string must be of the format *part:tag:flag:start:size* where *part* is the partition number, *tag* is the ID tag of the partition, *flag* is the set of permission flags, *start* is the starting sector number of the partition, and *size* is the number of blocks in the partition. See the description of the *datafile* below for more information on these fields.

-i Lets the command create the desired VTOC table, but prints the information to standard output instead of modifying the VTOC on the disk.

-m Automatically makes a file system on each partition that is: (1) mountable, (2) not read-only, and (3) marked as having a non-zero size.

-s *datafile*

The VTOC is populated according to a *datafile* created by the user. The *datafile* format is described below. This option causes all of the disk partition timestamp fields to be set to zero.

-n *volume_name*

Allows the disk to be given a *volume_name* up to 8 characters long.

-c *core_disk_type*

The three core disk configurations are: "0" (single disk configuration), "1" (first disk of a dual disk configuration), and "2" (second disk of a dual disk configuration).

Disk Configuration	Partition	Used For
0	0	root
	1	swap
	2	usr
1	0	root
	1	swap
	8	remaining space
2	2	usr

If no options are given, a default VTOC is created with partition 8 as the only mountable partition. Every VTOC generated by *fmthard* will also have partition 6 (the whole disk) and partition 7 (the boot partition).

The *datafile* contains one specification line for each partition, starting with partition 0. Each line is delimited by a new-line character (**\n**). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, separated by "white space" and having the following format.

```
partition tag flag starting_sector size_in_sectors
```

Where the entries have the following values.

<i>partition</i>	The partition number: 0-15 decimal or 0x0-0xf hexadecimal.
<i>tag</i>	The partition tag: a two-digit hex number. The following are reserved codes: V_BOOT 0x01, V_ROOT 0x02, V_SWAP 0x03, V_USR 0x04 and V_BACKUP 0x05.
<i>flag</i>	The flag allows a partition to be flagged as unmountable or read only, the masks being: V_UNMNT 0x01, and V_RDONLY 0x10.
<i>starting sector</i>	The sector number (decimal) on which the partition starts.
<i>size in sectors</i>	The number (decimal) of sectors occupied by the partition.

SEE ALSO

prtvtoc(1M).

WARNINGS

Special care should be exercised when overwriting an existing VTOC, as incorrect entries could result in current data being inaccessible.

After using *fmthard*(1M) on a bootable drive, you must execute *newboot*(1M) on that drive. *newboot*(1M) should also be executed after mirroring any partition on a bootable disk (executing either *sysadm rootsetup* or *sysadm mirror*).

If *newboot*(1M) is not executed after *fmthard*(1M), then the drive will become unbootable and may require a partial restore. Do not do a partial restore while *root* and */usr* are mirrored.

CAVEAT

When using the *-s* option, the user must allocate at least two sectors, beginning with sector 0, for the VTOC. This is normally designated as partition 7. Failure to allocate space for the VTOC may result in overwriting the VTOC, thereby destroying the disk partitioning information.

NAME

format – physically format a SCSI hard disk

SYNOPSIS

```
/etc/format [ -v ] [ -n ] /dev/rdisk/c?t?d?s6
```

DESCRIPTION

This command physically formats a Small Computer System Interface (SCSI) hard disk.

The super-user may use the *format* command in single-user state to prepare SCSI hard disks for use. The following options may be used with *format*:

-v verifies that the formatted SCSI hard disk is correct.

-n formatting is suppressed.

When a SCSI disk is formatted, a Physical Description (PD) Sector is placed at SCSI logical block zero.

SEE ALSO

Formatting in the *SCSI Operation Manual*.

CAVEAT

Should not be run in the background with other processes running. Cannot format non-SCSI hard disks.

DIAGNOSTICS

The format command exits with one of three values:

0 means NORMAL (or TRUE)

1 means execution errors

2 means bad command usage

WARNING

This command destroys any data that might be on the disk.



NAME

`frec` – recover files from a backup tape

SYNOPSIS

```
/etc/frec [-p path] [-f reqfile] raw_tape i_number:name ...
```

DESCRIPTION

`frec` recovers files from the specified *raw_tape* backup tape written by `volcopy(1M)` or `finc(1M)`, given their *i_numbers*. The data for each recovery request will be written into the file given by *name*.

The `-p` option allows you to specify a default prefixing *path* different from your current working directory. This will be prefixed to any *names* that are not fully qualified, i.e. that do not begin with `/` or `./`. If any directories are missing in the paths of recovery *names* they will be created.

`-p path` Specifies a prefixing *path* to be used to fully qualify any names that do not start with `/` or `./`.

`-f reqfile` Specifies a file which contains recovery requests. The format is *i_number:newname*, one per line.

EXAMPLES

To recover a file, i-number 1216 when backed-up, into a file named *junk* in your current working directory:

```
frec /dev/rSA/ctape1 1216:junk
```

To recover files with *i_numbers* 14156, 1232, and 3141 into files `/usr/src/cmd/a`, `/usr/src/cmd/b` and `/usr/joe/a.c`:

```
frec -p /usr/src/cmd /dev/rSA/ctape1 14156:a 1232:b
3141:/usr/joe/a.c
```

SEE ALSO

`ff(1M)`, `finc(1M)`, `labelit(1M)`.
`cpio(1)` in the *User's Reference Manual*.

BUGS

While paving a path (i.e. creating the intermediate directories contained in a pathname) `frec` can only recover inode fields for those directories contained on the tape and requested for recovery.



NAME

fsba – file system block analyzer

SYNOPSIS

```
/etc/fsba [ -b target_block_size ] file-system1 [ file-system2 ... ]
```

DESCRIPTION

The *fsba* command determines the disk space required to store the data from an existing file system in a new file system with the specified logical block size. Each *file-system* listed on the command line refers to an existing file system and should be specified by device name (e.g., */dev/rdisk/c1d0s2*).

The *target_block_size* specifies the logical block size in bytes of the new file system. Valid target block sizes are 512, 1024, and 2048. Default target block size is 1024. A block size of 2048 is supported only if the 2K file system package is installed.

The *fsba* command prints information about how many 512-byte disk sectors are allocated to store the data in the old (existing) file system and how many would be required to store the same data in a new file system with the specified logical block size. It also prints the number of allocated and free i-nodes for the existing file system.

If the number of free sectors listed for the new file system is negative, the data will not fit in the new file system unless the new file system is larger than the existing file system. The new file system must be made at least as large as the number of sectors listed by *fsba* as allocated for the new file system. The maximum size of the new file system is limited by the size of the disk partition used for the new file system.

Note that it is possible to specify a *target_block_size* that is smaller than the logical block size of the existing file system. In this case the new file system would require fewer sectors to store the data.

SEE ALSO

mkfs(1M), *prtvtoc(1M)*.



NAME

`fsck`, `dfscck` – check and repair file systems

SYNOPSIS

```
/etc/fsck [-y] [-n] [-sX] [-SX] [-t file] [-q] [-D] [-f] [-b] [file-systems]
/etc/dfscck [options1] fsys1 ... - [options2] fsys2 ...
```

DESCRIPTION

Fskck

`fsck` audits and interactively repairs inconsistent conditions for file systems. If the file system is found to be consistent, the number of files, blocks used, and blocks free are reported. If the file system is inconsistent the user is prompted for concurrence before each correction is attempted. It should be noted that most corrective actions will result in some loss of data. The amount and severity of data loss may be determined from the diagnostic output. The default action for each correction is to wait for the user to respond **yes** or **no**. If the user does not have write permission `fsck` defaults to a **-n** action.

The following options are accepted by `fsck`.

- y** Assume a **yes** response to all questions asked by `fsck`.
- n** Assume a **no** response to all questions asked by `fsck`; do not open the file system for writing.
- sX** Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-core copy of the superblock will not continue to be used, or written on the file system.

The **-sX** option allows for creating an optimal free-list organization.

If *X* is not given, the values used when the file system was created are used. The format of *X* is *cylinder size:gap size*.

- SX** Conditionally reconstruct the free list. This option is like **-sX** above except that the free list is rebuilt only if there were no discrepancies discovered in the file system. Using **-S** will force a **no** response to all questions asked by `fsck`. This option is useful for forcing free list reorganization on uncontaminated file systems.
- t** If `fsck` cannot obtain enough memory to keep its tables, it uses a scratch file. If the **-t** option is specified, the file named in the next argument is used as the scratch file, if needed. Without the **-t** flag, `fsck` will prompt the user for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when `fsck` completes.
- q** Quiet `fsck`. Do not print size-check messages. Unreferenced **ifos** will silently be removed. If `fsck` requires it, counts in the superblock will be automatically fixed and the free list salvaged.

- D Directories are checked for bad blocks. Useful after system crashes.
- f Fast check. Check block and sizes and check the free list. The free list will be reconstructed if it is necessary.
- b Reboot. If the file system being checked is the root file system and modifications have been made, then either remount the root file system or reboot the system. A remount is done only if there was minor damage.

If no *file-systems* are specified, *fsck* will read a list of default file systems from the file */etc/checklist*.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one i-node or the free list.
2. Blocks claimed by an i-node or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
 - Incorrect number of blocks.
 - Directory size not 16-byte aligned.
5. Bad i-node format.
6. Blocks not accounted for anywhere.
7. Directory checks:
 - File pointing to unallocated i-node.
 - I-node number out of range.
8. Super Block checks:
 - More than 65536 i-nodes.
 - More blocks for i-nodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free i-node count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the user's concurrence, reconnected by placing them in the **lost+found** directory, if the files are nonempty. The user will be notified if the file or directory is empty or not. Empty files or directories are removed, as long as the **-n** option is not specified. *fsck* will force the reconnection of nonempty directories. The name assigned is the i-node number. The only restriction is that the directory **lost+found** must preexist in the root of the file system being checked and must have empty slots in which entries can be made. This is accomplished by making **lost+found**, copying a number of files to the directory, and then removing them (before *fsck* is executed).

Checking the raw device is almost always faster and should be used with everything but the *root* file system.

Dfsck

This version of the *fsck* command is appropriate for 3B2 computers only if equipped with dual hard disk drives. *Dfsck* should not be used to check the *root* file system.

Dfsck allows two file system checks on two different drives simultaneously. *options1* and *options2* are used to pass options to *fsck* for the two sets of file systems. A **-** is the separator between the file system groups.

The *dfsck* program permits a user to interact with two *fsck* programs at once. To aid in this, *dfsck* will print the file system name for each message to the user. When answering a question from *dfsck*, the user must prefix the response with a 1 or a 2 (indicating that the answer refers to the first or second file system group).

FILES

`/etc/checklist` contains default list of file systems to check.

SEE ALSO

checkfsys(1M), *crash(1M)*, *mkfs(1M)*, *ncheck(1M)*, *uadmin(1M)*, *checklist(4)*, *fs(4)*, *uadmin(2)* in the *Programmer's Reference Manual*.

BUGS

I-node numbers for `.` and `..` in each directory are not checked for validity.



NAME

fsdb – file system debugger

SYNOPSIS

/etc/fsdb special [-]

DESCRIPTION

fsdb can be used to patch up a damaged file system after a crash. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an i-node. These greatly simplify the process of correcting control block entries or descending the file system tree.

fsdb contains several error-checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking *fsdb* with the optional - argument or by the use of the O symbol. (*fsdb* reads the i-size and f-size entries from the superblock of the file system as the basis for these checks.)

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

fsdb reads a block at a time and will therefore work with raw as well as block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The symbols recognized by *fsdb* are:

#	absolute address
i	convert from i-number to i-node address
b	convert to block address
d	directory slot offset
+, -	address arithmetic
q	quit
>, <	save, restore an address
=	numerical assignment
=+	incremental assignment
=-	decremental assignment
='	character string assignment
O	error checking flip flop
p	general print facilities
f	file print facility
B	byte mode
W	word mode
D	double word mode
!	escape to shell

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the delete character. If a

number follows the **p** symbol, that many entries are printed. A check is made to detect block boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed. The print options available are:

i	print as i-nodes
d	print as directories
o	print as octal words
e	print as decimal words
c	print as characters
b	print as octal bytes

The **f** symbol is used to print data blocks associated with the current i-node. If followed by a number, that block of the file is printed. (Blocks are numbered from zero.) The desired print option letter follows the block number, if present, or the **f** symbol. This print facility works for small as well as large files. It checks for special devices and that the block pointers used to find the data are not zero.

Dots, tabs, and spaces may be used as function delimiters but are not necessary. A line with just a new-line character will increment the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing the user to step through a region of a file system. Information is printed in a format appropriate to the data type. Bytes, words and double words are displayed with the octal address followed by the value in octal and decimal. A **.B** or **.D** is appended to the address for byte and double word values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. I-nodes are printed with labeled fields describing each element.

The following mnemonics are used for i-node examination and refer to the current working i-node:

md	mode
ln	link count
uid	user ID number
gid	group ID number
sz	file size
a#	data block numbers (0 - 12)
at	access time
mt	modification time
maj	major device number
min	minor device number

EXAMPLES

386i	prints i-number 386 in an i-node format. This now becomes the current working i-node.
ln=4	changes the link count for the working i-node to 4.
ln+=1	increments the link count by 1.
fc	prints, in ASCII, block zero of the file associated with the working i-node.

- 2i.fd prints the first 32 directory entries for the root i-node of this file system.
- d5i.fc changes the current i-node to that associated with the 5th directory entry (numbered from zero) found from the above command. The first logical block of the file is then printed in ASCII.
- 512B.p0o prints the superblock of this file system in octal.
- 2i.a0b.d7=3 changes the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.
- d7.nm="name" changes the name field in the directory slot to the given string. Quotes are optional when used with nm if the first character is alphabetic.
- a2b.p0d prints the third block of the current i-node as directory entries.

SEE ALSO

fsck(1M), dir(4), fs(4).



NAME

fsstat – report file system status

SYNOPSIS

/etc/fsstat *special_file*

DESCRIPTION

fsstat reports on the status of the file system on *special_file*. During startup, this command is used to determine if the file system needs checking before it is mounted. *fsstat* succeeds if the file system is unmounted and appears okay. For the root file system, it succeeds if the file system is active and not marked bad.

SEE ALSO

fs(4).

DIAGNOSTICS

The command has the following exit codes:

- 0 -- the file system is not mounted and appears okay,
(except for root where 0 means mounted and okay).
- 1 -- the file system is not mounted and needs to be checked.
- 2 -- the file system is mounted.
- 3 -- the command failed.



NAME

fstyp – determine file system identifier

SYNOPSIS

fstyp special

DESCRIPTION

fstyp allows the user to determine the file system identifier of mounted or unmounted file systems using heuristic programs. The file system type is required by *mount(2)* and sometimes by *mount(1M)* to mount file systems of different types.

The directory */etc/fstyp.d* contains a program for each file system type to be checked; each of these programs applies some appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file-system identifier for that type and exits with a return code of 0; otherwise it prints error messages on standard error and exits with a non-zero return code. *fstyp* runs the programs in */etc/fstyp.d* in alphabetical order, passing *special* as an argument; if any program succeeds, its file-system type identifier is printed and *fstyp* exits immediately. If no program succeeds, *fstyp* prints "Unknown_fstyp" to indicate failure.

WARNING

The use of heuristics implies that the result of *fstyp* is not guaranteed to be accurate.

SEE ALSO

mount(1M).

mount(2), *sysfs(2)* in the *Programmer's Reference Manual*.



NAME

fumount – forced unmount of an advertised resource

SYNOPSIS

fumount [-w *sec*] *resource*

DESCRIPTION

fumount unadvertises *resource* and disconnects remote access to the resource. The *-w sec* causes a delay of *sec* seconds prior to the execution of the disconnect.

When the forced unmount occurs, an administrative shell script is started on each remote computer that has the resource mounted (`/usr/bin/rfuadmin`). If a grace period of seconds is specified, `rfuadmin` is started with the `fuwarn` option. When the actual forced unmount is ready to occur, `rfuadmin` is started with the `fumount` option. See the `rfuadmin(1M)` man page for information on the action taken in response to the forced unmount.

This command is restricted to the super-user.

ERRORS

If *resource* (1) does not physically reside on the local machine, (2) is an invalid resource name, (3) is not currently advertised and is not remotely mounted, or (4) the command is not run with super-user privileges, an error message will be sent to standard error.

SEE ALSO

`adv(1M)`, `mount(1M)`, `rfuadmin(1M)`, `rfudaemon(1M)`, `rmount(1M)`, `unadv(1M)`.



NAME

fusage – disk access profiler

SYNOPSIS

fusage *[[mount_point] | [advertised_resource] | [block_special_device] [...]]*

DESCRIPTION

When used with no options, **fusage** reports block i/o transfers, in kilobytes, to and from all locally mounted file systems and advertised Remote File Sharing resources on a per client basis. The count data are cumulative since the time of the mount. When used with an option, **fusage** reports on the named file system, advertised resource, or block special device.

The report includes one section for each file system and advertised resource and has one entry for each machine that has the directory remotely mounted, ordered by decreasing usage. Sections are ordered by device name; advertised resources that are not complete file systems will immediately follow the sections for the file systems they are in.

SEE ALSO

adv(1M), mount(1M), df(1M), crash(1M).



NAME

fuser – identify processes using a file or file structure

SYNOPSIS

/etc/fuser [-ku] files | resources [-] [[-ku] files | resources]

DESCRIPTION

fuser outputs the process IDs of the processes that are using the *files* or remote *resources* specified as arguments. Each process ID is followed by a letter code, interpreted as follows: if the process is using the file as 1) its current directory, the code is *c*, 2) the parent of its current directory (only when the file is being used by the system), the code is *p*, or 3) its root directory, the code is *r*. For block special devices with mounted file systems, all processes using any file on that device are listed. For remote resource names, all processes using any file associated with that remote resource (Remote File Sharing) are reported. (*fuser* cannot use the mount point of the remote resource; it must use the resource name.) For all other types of files (text files, executables, directories, devices, etc.) only the processes using that file are reported.

The following options may be used with *fuser*:

- u the user login name, in parentheses, also follows the process ID.
- k the SIGKILL signal is sent to each process. Since this option spawns kills for each process, the kill messages may not show up immediately [see *kill(2)*].

If more than one group of files are specified, the options may be respecified for each additional group of files. A lone dash cancels the options currently in force; then, the new set of options applies to the next group of files.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

You cannot list processes using a particular file from a remote resource mounted on your machine. You can only use the resource name as an argument.

Any user with permission to read */dev/kmem* and */dev/mem* can use *fuser*. Only the super-user can terminate another user's process

FILES

/unix for system namelist
/dev/kmem for system image
/dev/mem also for system image

SEE ALSO

mount(1M).
ps(1) in the *User's Reference Manual*.
kill(2), *signal(2)* in the *Programmer's Reference Manual*.



NAME

gencc – create a front-end to the *cc* command

SYNOPSIS

gencc

DESCRIPTION

The *gencc* command is an interactive command designed to aid in the creation of a front-end to the *cc* command. Since hard-coded pathnames have been eliminated from the C Compilation System (CCS), it is possible to move pieces of the CCS to new locations without recompiling the CCS. The new locations of moved pieces can be specified through the *-Y* option to the *cc* command. However, it is inconvenient to supply the proper *-Y* options with every invocation of the *cc* command. Further, if a system administrator moves pieces of the CCS, such movement should be invisible to users.

The front-end to the *cc* command which *gencc* generates is a one-line shell script which calls the *cc* command with the proper *-Y* options specified. The front-end to the *cc* command will also pass all user supplied options to the *cc* command.

gencc prompts for the location of each tool and directory which can be respecified by a *-Y* option to the *cc* command. If no location is specified, it assumes that that piece of the CCS has not been relocated. After all the locations have been prompted for, *gencc* will create the front-end to the *cc* command.

gencc creates the front-end to the *cc* command in the current working directory and gives the file the same name as the *cc* command. Thus, *gencc* can not be run in the same directory containing the actual *cc* command. Further, if a system administrator has redistributed the CCS, the actual *cc* command should be placed somewhere which is not typically in a user's PATH (e.g., */lib*). This will prevent users from accidentally invoking the *cc* command without using the front-end.

CAVEATS

gencc does not produce any warnings if a tool or directory does not exist at the specified location. Also, *gencc* does not actually move any files to new locations.

FILES

./cc front-end to *cc*

SEE ALSO

cc(1).



NAME

`getmajor` -- print major number(s) of hardware devices

SYNOPSIS

`/etc/getmajor name | ID_code`

DESCRIPTION

The `getmajor` command prints all major numbers for the requested device found in the system Equipped Device Table (EDT). Slot and major numbers are the same for boards that are installed directly into the backplane slots of the computer. `ID_code` is a number between 0x0 and 0xffff that a device uses to identify itself.

Devices that are on extended buses (e.g., Small Computer System Interface (SCSI) target controllers) do not have board ID codes. The proper way to use `getmajor` with these devices is `/etc/getmajor name`.

DIAGNOSTICS

If successful, a zero is returned. If `name` or `ID_code` is not found, a blank line is printed and the return code is nonzero.

SEE ALSO

`edittbl(1M)`, `prtconf(1M)`.



NAME

getty – set terminal type, modes, speed, and line discipline

SYNOPSIS

```
/etc/getty [ -h ] [ -t timeout ] line [ speed [ type [ linedisc ] ] ]
/etc/getty -c file
```

DESCRIPTION

getty is a program that is invoked by *init*(1M). It is the second process in the series, (*init-getty-login-shell*) that ultimately connects a user with the UNIX system. It can only be executed by the super-user; that is, a process with the user-ID of *root*. Initially *getty* prints the login message field for the entry it is using from */etc/gettydefs*. *getty* reads the user's login name and invokes the *login*(1) command with the user's name as argument. While reading the name, *getty* attempts to adapt the system to the speed and type of terminal being used. It does this by using the options and arguments specified.

Line is the name of a tty line in */dev* to which *getty* is to attach itself. *getty* uses this string as the name of a file in the */dev* directory to open for reading and writing. Unless *getty* is invoked with the *-h* flag, *getty* will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed. The *-t* flag plus *timeout* (in seconds), specifies that *getty* should exit if the open on the line succeeds and no one types anything in the specified number of seconds.

Speed, the optional second argument, is a label to a speed and tty definition in the file */etc/gettydefs*. This definition tells *getty* at what speed to initially run, what the login message should look like, what the initial tty settings are, and what speed to try next should the user indicate that the speed is inappropriate (by typing a *<break>* character). The default *speed* is 300 baud.

Type, the optional third argument, is a character string describing to *getty* what type of terminal is connected to the line in question. *getty* recognizes the following types:

none	default
ds40-1	Dataspeed40/1
tektronix,tek	Tektronix
vt61	DEC vt61
vt100	DEC vt100
hp45	Hewlett-Packard 45
c100	Concept 100

The default terminal is **none**; i.e., any crt or normal terminal unknown to the system. Also, for terminal type to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled in the default condition.

Linedisc, the optional fourth argument, is a character string describing which line discipline to use in communicating with the terminal. Again the hooks for line disciplines are available in the operating system but there is only one presently available, the default line discipline, LDISC0.

When given no optional arguments, *getty* sets the *speed* of the interface to 300 baud, specifies that raw mode is to be used (awaken on every character), that

echo is to be suppressed, either parity allowed, new-line characters will be converted to carriage return-line feed, and tab expansion performed on the standard output. It types the login message before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pushing the "break" key. This will cause *getty* to attempt the next *speed* in the series. The series that *getty* tries is determined by what it finds in */etc/gettydefs*.

After the user's name has been typed in, it is terminated by a new-line or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see *ioctl(2)*).

The user's name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is non-empty, the system is told to map any future upper-case characters into the corresponding lower-case characters.

Finally, *login* is *exec'd* with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to *login*, which will place them in the environment (see *login(1)*).

A check option is provided. When *getty* is invoked with the *-c* option and *file*, it scans the file as if it were scanning */etc/gettydefs* and prints out the results to the standard output. If there are any unrecognized modes or improperly constructed entries, it reports these. If the entries are correct, it prints out the values of the various flags. See *ioctl(2)* to interpret the values. Note that some values are added to the flags automatically.

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

ct(1C), *init(1M)*, *gettydefs(4)*, *inittab(4)*, *tty(7)*.
login(1) in the *User's Reference Manual*.
ioctl(2) in the *Programmer's Reference Manual*.

BUGS

While *getty* understands simple single character quoting conventions, it is not possible to quote certain special control characters used by *getty*. Thus, you cannot login via *getty* and type a #, @, /, !, _ backspace, ^U, ^D, or & as part of your login name or arguments. *getty* uses them to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. They will always be interpreted as having their special meaning.

NAME

`hdeadd` — add/delete `hdelog` (Hard Disk Error Log) reports

SYNOPSIS

```

/etc/hdeadd -a [ options ]
/etc/hdeadd -d [ options ]
/etc/hdeadd -e [ [ -D ] major minor ]
/etc/hdeadd -f filename
/etc/hdeadd -r [ -D ] major minor filename
/etc/hdeadd -s [ -D ] major minor filename

```

DESCRIPTION

This command is part of the bad block handling utility. It may be used only by the super-user for manually adding or deleting disk error reports recorded by `hdelogger`. These include disk errors reported while in firmware mode and disk errors that cause the system to PANIC.

`hdeadd` may be used to print the list of equipped disks or to determine if a specific disk device is on the list. In addition, this command has some options that are for use in testing the feature.

The following options may be used with `hdeadd`:

- `a` `hdeadd` allows a Hard Disk Error (HDE) report to be added manually to the HDE Log of a disk.
- `d` `hdeadd` allows a specific report or a range of reports to be deleted from the HDE Log of a disk.
- `e` prints out the list of major/minor device numbers of the equipped hard disks. If the *major* and *minor* device numbers are also provided, it determines if that specification is an equipped hard disk. The result is both printed on the standard output and is used to determine the exit status. A NORMAL (or TRUE) exit means it is an equipped disk.
- `f` the file specified by *filename* is assumed to contain a canned set of HDE Log manipulations. Each line of text contains one specification in the command argument form, starting with a `-a` or a `-d` option.
- `s` saves a copy of the HDE Log of the specified (by *major/minor* device number) disk in the file specified by *filename*.
- `r` restores the HDE Log of the specified disk from the file specified by *filename*.

The valid *options* are only *hard disk error* specifications.

The valid *options* are either a *hard disk error* specification or an *error range* specification.

A *hard disk error* specification includes the following values:

- `D` *maj min* Specifies the major device number (*maj*) and minor device number (*min*) of the disk.

- b** *blockno* **Normal form:** Specifies the physical disk block number in integer counter form (i.e., treating the disk as a simple stream of blocks). Physical disk block numbering starts with zero meaning sector 0 of track 0 of cylinder 0. This is the normal form that is reported by the operating system.
- B** *cyl trk sec* **Alternate form:** Specifies the physical disk block number in terms of its physical cylinder number (*cyl*), track number within cylinder (*trk*), and sector number within track (*sec*). This alternate form is available to cover the possibility of a non-operating system detector reporting block numbers in this hardware form.
- t** *mmddhhmm[yy]* **Optional:** Specifies the time of day when the error actually occurred. If omitted when adding reports, the current time is used. If omitted when deleting reports, any reports for the given block are deleted.

An *error range* specification includes the following values:

- D** *maj min* Specifies the major device number (*maj*) and minor device number (*min*) of the disk.
- F** *mmddhhmm[yy]* **Optional:** Specifies the "from" time for the time interval being purged. If omitted, zero (the beginning of time) is used.
- T** *mmddhhmm[yy]* **Optional:** Specifies the "to" time for the time interval being purged. If omitted, the end of time is used. The range comparisons include the end values of the range in the purge.

FILES

/dev/hdelog

SEE ALSO

hdefix(1M), hdelogger(1M), hdelog(7).
Bad Block Handling in the *System Administrator's Guide*.

DIAGNOSTICS

The HDE commands exit with one of three values:

- 0 means NORMAL, or TRUE
- 1 means bad command usage or execution errors
- 2 means BAD BLOCKS or FALSE (but command executed successfully)

NAME

`hdefix` – report or change bad block mapping on a hard disk device

SYNOPSIS

```
/etc/hdefix -p [ -D major minor ]
/etc/hdefix -a [ -D major minor [ -b blockno ... ] ]
/etc/hdefix -a [ -D major minor [ -B cyl trk sec ... ] ]
```

DESCRIPTION

The `hdefix` command is part of the bad block handling utility. This command maps bad blocks to surrogate images in an area not accessible by the user.

Before attempting to execute `hdefix`, the system must be brought to the single-user state using the command 'init s'. Only super-user can use `hdefix` to print a list of blocks currently mapped to surrogate images on the equipped hard disk devices or to change the mapping of these blocks.

When the mapping to surrogate images is changed, block initialization is performed. If the original block can be read, its data is written to the new surrogate image to prevent data loss. If the original block is unreadable, zeros are written to the new surrogate image. This will usually result in some data loss.

If the block is associated with a file system, the file system may be damaged as a result of the mapping change. To handle this situation, the file system is marked dirty, which means `fsck(1M)` must be run before the file system can be used, and a system reboot is forced after all other bad block processing is complete. If the block is a data block of a file, that file will be corrupted, even after this recovery has finished.

The following options may be used with `hdefix`:

- p** prints a report that shows both the functional blocks and currently mapped bad blocks. If a specific hard disk device is specified (by giving its *major* and *minor* device numbers), only the report for that hard disk device is printed. If no particular hard disk device is specified, a report is given for each equipped disk.
- D** used to specify the *major* device number and *minor* device number of a hard disk device.
- a** used to map new bad blocks. If no arguments follow the **-a** option, all equipped hard disk devices are processed, using the HDE Log on each hard disk device to determine which blocks to map. If a specific hard disk device is specified, only that disk device is processed. If one or more block numbers are specified, those blocks are mapped, instead of using the block numbers listed in HDE Log. This is the only way to map an unreadable block containing the HDE Log.
- b *blockno*** specifies the physical hard disk block number. Physical hard disk block numbering starts with zero, meaning block (sector) 0 of track 0 of cylinder 0. The *blockno*

value ranges from block number 0 through the maximum number of blocks on a particular hard disk drive minus 1.

-B *cyl trk sec*

specifies the physical disk block number in terms of its physical cylinder number (*cyl*), track number within cylinder (*trk*), and sector (block) number within track (*sec*). This alternate form is available for reporting bad block data obtained without using the normal system capabilities (e.g., off-line diagnostics provided by the manufacturer). This option is not supported on SCSI disks.

FILES

/dev/hdelog

SEE ALSO

fsck(1M), init(1M), hdeadd(1M), hdelogger(1M), hdelog(7).
Bad Block Handling in the *System Administrator's Guide*.

RETURN CODES

The *hdefix* command exits with one of three values:

- 0 means NORMAL (or TRUE)
- 1 means bad command usage or execution errors
- 2 means BAD BLOCKS (or FALSE) (but command executed successfully)

NAME

hdelogger – Hard Disk Error status report command and Log Daemon

SYNOPSIS

/etc/hdelogger [-s] [-f] [-D maj min]

DESCRIPTION

This command is part of the bad block handling utility. It is executed automatically by the *init* in run levels 2, 3 and 4.

The *hdelogger* command serves two purposes. When run by the *init* process (process 1 – see *init*(1M)), this command performs the functions of the Hard Disk Error (HDE) Log Daemon. These functions include providing summaries of outstanding errors during system startup and shutdown transitions, along with adding new errors to HDE Logs and giving the revised status summaries as errors are reported by hard disk drivers. When run as the daemon, no options are used.

When run as a normal command (process 1 is not its parent), this command provides on the spot reports of outstanding errors as recorded in the HDE Logs of equipped hard disks. You must be the super-user to run the command this way. The following options control report generation:

- s** Specifies that summary reports are to be generated. The summary report provides sufficient information for normal bad block handling operations. This is the default.
- f** Specifies that full reports are to be generated. This is intended mainly for testing the bad block handling feature, but is available in case additional detail is needed for troubleshooting complicated problems.
- D maj min** Restricts the report generation to a specific hard disk. If this option is omitted, reports will be generated for all equipped hard disks.

FILES

/dev/hdelog

SEE ALSO

hdeadd(1M), hdefix(1M), hdelog(7).
Bad Block Handling, in the *System Administrator's Guide*.

DIAGNOSTICS

The HDE commands exit with one of three values:

- 0 means NORMAL, or TRUE
- 1 means bad command usage or execution errors
- 2 means BAD BLOCKS or FALSE (but command executed successfully)



NAME

`id` – print user and group IDs and names

SYNOPSIS

`id`

DESCRIPTION

`id` outputs the user and group IDs and the corresponding names of the invoking process. If the effective and real IDs are different, both are printed.

SEE ALSO

`logname(1)` in the *User's Reference Manual*.

`getuid(2)` in the *Programmer's Reference Manual*.



NAME

idload – Remote File Sharing user and group mapping

SYNOPSIS

```
idload [-n] [-g g_rules] [-u u_rules] [directory]
idload -k
```

DESCRIPTION

idload is used on Remote File Sharing server machines to build translation tables for user and group ids. It takes your `/etc/passwd` and `/etc/group` files and produces translation tables for user and group ids from remote machines, according to the rules set down in the *u_rules* and *g_rules* files. If you are mapping by user and group name, you will need copies of remote `/etc/passwd` and `/etc/group` files. If no rules files are specified, remote user and group ids are mapped to MAXUID+1 (this is an id number that is one higher than the highest number you could assign on your system.)

By default, the remote password and group files are assumed to reside in `/usr/nserve/auth.info/domain/nodename/[passwd|group]`. The *directory* argument indicates that some directory structure other than `/usr/nserve/auth.info` contains the *domain/nodename* `passwd` and `group` files. (*nodename* is the name of the computer the files are from and *domain* is the domain that computer is a member of.)

You must run **idload** to put the mapping into place. Global mapping will take effect immediately for machines that have one of your resources currently mounted. Mapping for other specific machines will take effect when each machine mounts one of your resources.

- n This is used to do a trial run of the id mapping. No translation table will be produced, however, a display of the mapping is output to the terminal (*stdout*).
- k This is used to print the idmapping that is currently in use. (Specific mapping for remote machines will not be shown until that machine mounts one of your resources.)
- u *u_rules* The *u_rules* file contains the rules for user id translation. The default rules file is `/usr/nserve/auth.info/uid.rules`.
- g *g_rules* The *g_rules* file contains the rules for group id translation. The default rules file is `/usr/nserve/auth.info/gid.rules`.

This command is restricted to the super-user.

Rules

The rules files have two types of sections (both optional): **global** and **host**. There can be only one global section, though there can be one host section for each computer you want to map.

The **global** section describes the default conditions for translation for any machines that are not explicitly referenced in a **host** section. If the global section is missing, the default action is to map all remote user and group ids from undefined computers to MAXUID+1. The syntax of the first line of the **global** section is:

global

A **host** section is used for each machine or group of machines that you want to map differently from the global definitions. The syntax of the first line of each **host** section is:

host *name* ...

where *name* is replaced by the full name of a computer (*domain.nodename*).

The format of a rules file is described below. (All lines are optional, but must appear in the order shown.)

global

default *local* | **transparent**

exclude *remote_id-remote_id* | *remote_id*

map *remote_id:local*

host *domain.nodename* [*domain.nodename...*]

default *local* | **transparent**

exclude *remote_id-remote_id* | *remote_id* | *remote_name*

map *remote:local* | *remote* | **all**

Each of these instruction types is described below.

The line

default *local* | **transparent**

defines the mode of mapping for remote users that are not specifically mapped in instructions in other lines. **transparent** means that each remote user and group id will have the same numeric value locally unless it appears in the **exclude** instruction. *local* can be replaced by a local user name or id to map all users into a particular local name or id number. If the default line is omitted, all users that are not specifically mapped are mapped into a "special guest" login id.

The line

exclude *remote_id-remote_id* | *remote_id* | *remote_name*

defines remote ids that will be excluded from the **default** mapping. The **exclude** instruction must precede any **map** instructions in a block. You can use a range of id numbers, a single id number, or a single name. (*remote_name* cannot be used in a *global* block.)

The line

map *remote:local* | *remote* | **all**

defines the local ids and names that remote ids and names will be mapped into. *remote* is either a remote id number or remote name; *local* is either a local id number or local name. Placing a colon between a *remote* and a *local* will give the value on the left the permissions of the value on the right. A single *remote* name or id will assign the user or group permissions of the same local

name or id. **all** is a predefined alias for the set of all user and group ids found in the local `/etc/passwd` and `/etc/group` files. (You cannot map by remote name in **global** blocks.)

NOTE: **idload** will always output warning messages for **map all**, since password files always contain multiple administrative user names with the same id number. The first mapping attempt on the id number will succeed, each subsequent attempts will produce a warning.

Remote File Sharing doesn't need to be running to use **idload**.

EXIT STATUS

On successful completion, **idload** will produce one or more translation tables and return a successful exit status. If **idload** fails, the command will return an exit status of zero and not produce a translation table.

ERRORS

If (1) either rules file cannot be found or opened, (2) there are syntax errors in the rules file, (3) there are semantic errors in the rules file, (4) **host** password or group information could not be found, or (5) the command is not run with super-user privileges, an error message will be sent to standard error. Partial failures will cause a warning message to appear, though the process will continue.

FILES

`/etc/passwd`
`/etc/group`
`/usr/nserve/auth.info/domain/nodename/[user|group]`
`/usr/nserve/auth.info/uid.rules`
`/usr/nserve/auth.info/gid.rules`

SEE ALSO

`mount(1M)`.

"Remote File Sharing" chapter of the *System Administrator's Guide* for detailed information on ID mapping.



NAME

infocmp – compare or print out terminfo descriptions

SYNOPSIS

infocmp [-d] [-c] [-n] [-I] [-L] [-C] [-r] [-u] [-s d|i|l|c] [-v] [-V] [-1]
[-w width] [-A directory] [-B directory] [termname ...]

DESCRIPTION

infocmp can be used to compare a binary *terminfo(4)* entry with other terminfo entries, rewrite a *terminfo(4)* description to take advantage of the *use=* terminfo field, or print out a *terminfo(4)* description from the binary file (*term(4)*) in a variety of formats. In all cases, the boolean fields will be printed first, followed by the numeric fields, followed by the string fields.

Default Options

If no options are specified and zero or one *termnames* are specified, the *-I* option will be assumed. If more than one *termname* is specified, the *-d* option will be assumed.

Comparison Options [-d] [-c] [-n]

infocmp compares the *terminfo(4)* description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termnames*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: F for boolean variables, *-1* for integer variables, and NULL for string variables.

- d* produce a list of each capability that is different. In this manner, if one has two entries for the same terminal or similar terminals, using *infocmp* will show what is different between the two entries. This is sometimes necessary when more than one person produces an entry for the same terminal and one wants to see what is different between the two.
- c* produce a list of each capability that is common between the two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the *-u* option is worth using.
- n* produce a list of each capability that is in neither entry. If no *termnames* are given, the environment variable *TERM* will be used for both of the *termnames*. This can be used as a quick check to see if anything was left out of the description.

Source Listing Options [-I] [-L] [-C] [-r]

The *-I*, *-L*, and *-C* options will produce a source listing for each terminal named.

- I* use the *terminfo(4)* names
- L* use the long C variable name listed in *<term.h>*
- C* use the *termcap* names
- r* when using *-C*, put out all capabilities in *termcap* form

If no *termnames* are given, the environment variable *TERM* will be used for the terminal name.

The source produced by the `-C` option may be used directly as a *termcap* entry, but not all of the parameterized strings may be changed to the *termcap* format. *infocmp* will attempt to convert most of the parameterized information, but that which it doesn't will be plainly marked in the output and commented out. These should be edited by hand.

All padding information for strings will be collected together and placed at the beginning of the string where *termcap* expects it. Mandatory padding (padding information with a trailing '/') will become optional.

All *termcap* variables no longer supported by *terminfo(4)*, but which are derivable from other *terminfo(4)* variables, will be output. Not all *terminfo(4)* capabilities will be translated; only those variables which were part of *termcap* will normally be output. Specifying the `-r` option will take off this restriction, allowing all capabilities to be output in *termcap* form.

Note that because padding is collected to the beginning of the capability, not all capabilities are output, mandatory padding is not supported, and *termcap* strings were not as flexible, it is not always possible to convert a *terminfo(4)* string capability into an equivalent *termcap* format. Not all of these strings will be able to be converted. A subsequent conversion of the *termcap* file back into *terminfo(4)* format will not necessarily reproduce the original *terminfo(4)* source.

Some common *terminfo* parameter sequences, their *termcap* equivalents, and some terminal types which commonly have such sequences, are:

Terminfo	Termcap	Representative Terminals
%p1%c	%.	adm
%p1%d	%d	hp, ANSI standard, vt100
%p1%'x'%'y'%'z'	%+x	concept
%i	%i	ANSI standard, vt100
%p1%'?'%'x'%'>'%'t%'p1%'y'%'>';	%>xy	concept
%p2 is printed before %p1	%r	hp

Use= Option [-u]

`-u` produce a *terminfo(4)* source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals *termnames*. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with `use=` fields for the other terminals. In this manner, it is possible to retrofit generic *terminfo* entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times or by different people so that each description is a full description, using *infocmp* will show what can be done to change one description to be relative to the other.

A capability will get printed with an at-sign (@) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value gets printed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries

that has this capability gives a different value for the capability than that in the first *termname*.

The order of the other *termname* entries is significant. Since the terminfo compiler *tic*(1M) does a left-to-right scan of the capabilities, specifying two **use=** entries that contain differing entries for the same capabilities will produce different results depending on the order that the entries are given in. *infocmp* will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability *after* a **use=** entry that contains that capability will cause the second specification to be ignored. Using *infocmp* to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying extra **use=** fields that are superfluous. *infocmp* will flag any other *termname* **use=** fields that were not needed.

Other Options [-s d|i|l|c] [-v] [-V] [-1] [-w width]

- s sort the fields within each type according to the argument below:
 - d leave fields in the order that they are stored in the *terminfo* database.
 - i sort by *terminfo* name.
 - l sort by the long C variable name.
 - c sort by the *termcap* name.
- If no **-s** option is given, the fields printed out will be sorted alphabetically by the *terminfo* name within each type, except in the case of the **-C** or the **-L** options, which cause the sorting to be done by the *termcap* name or the long C variable name, respectively.
- v print out tracing information on standard error as the program runs.
 - V print out the version of the program in use on standard error and exit.
 - 1 cause the fields to be printed out one to a line. Otherwise, the fields will be printed several to a line to a maximum width of 60 characters.
 - w change the output to width characters.

Changing Databases [-A directory] [-B directory]

The location of the compiled *terminfo*(4) database is taken from the environment variable **TERMINFO**. If the variable is not defined, or the terminal is not found in that location, the system *terminfo*(4) database, usually in */usr/lib/terminfo*, will be used. The options **-A** and **-B** may be used to override this location. The **-A** option will set **TERMINFO** for the first *termname* and the **-B** option will set **TERMINFO** for the other *termnames*. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people. Otherwise the terminals would have to be named differently in the *terminfo*(4) database for a comparison to be made.

FILES

/usr/lib/terminfo/??/* compiled terminal description database

DIAGNOSTICS

malloc is out of space!

There was not enough memory available to process all the terminal descriptions requested. Run *infocmp* several times, each time including a subset of the desired *termnames*.

use= order dependency found:

A value specified in one relative terminal specification was different from that in another relative terminal specification.

'use=*term*' did not add anything to the description.

A relative terminal name did not contribute anything to the final description.

must have at least two terminal names for a comparison to be done.

The *-u*, *-d* and *-c* options require at least two terminal names.

SEE ALSO

captainfo(1M), *tic(1M)*, *curses(3X)*, *term(4)*, *terminfo(4)* in the *System Administrator's Reference Manual*.

Chapter 10 of the *Programmer's Guide*.

NOTE

The *termcap* database (from earlier releases of UNIX System V) may not be supplied in future releases.

NAME

init, telinit – process control initialization

SYNOPSIS

`/etc/init [0123456SsQqabc]`

`/etc/telinit [0123456SsQqabc]`

DESCRIPTION

Init

init is a general process spawner. Its primary role is to create processes from information stored in the file `/etc/inittab` (see *inittab(4)*).

At any given time, the system is in one of eight possible run levels. A run level is a software configuration of the system under which only a selected group of processes exist. The processes spawned by *init* for each of these run levels is defined in `/etc/inittab`. *init* can be in one of eight run levels, **0–6** and **S** or **s** (run levels **S** and **s** are identical). The run level changes when a privileged user runs `/etc/init`. This user-spawned *init* sends appropriate signals to the original *init* spawned by the operating system when the system was booted, telling it which run level to change to.

The following are the arguments to *init*.

- 0** shut the machine down so it is safe to remove the power. Have the machine remove power if it can.
- 1** put the system in single-user mode. Unmount all file systems except root. All user processes are killed except those connected to the console.
- 2** put the system in multi-user mode. All multi-user environment terminal processes and daemons are spawned. This state is commonly referred to as the multi-user state.
- 3** start the remote file sharing processes and daemons. Mount and advertise remote resources. Run level **3** extends multi-user mode and is known as the remote-file-sharing state.
- 4** is available to be defined as an alternative multi-user environment configuration. It is not necessary for system operation and is usually not used.
- 5** Stop the UNIX system and go to the firmware monitor.
- 6** Stop the UNIX system and reboot to the state defined by the `initdefault` entry in `/etc/inittab`.
- a,b,c** process only those `/etc/inittab` entries having the **a**, **b** or **c** run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.
- Q,q** re-examine `/etc/inittab`.
- S,s** enter single-user mode. When this occurs, the terminal which executed this command becomes the system console. This is the only run level that doesn't require the existence of a

properly formatted `/etc/inittab` file. If this file does not exist, then by default the only legal run level that `init` can enter is the single-user mode. When the system enters `S` or `s`, all mounted file systems remain mounted and only processes spawned by `init` are killed.

When a UNIX system is booted, `init` is invoked and the following occurs. First, `init` looks in `/etc/inittab` for the `initdefault` entry (see `inittab(4)`). If there is one, `init` uses the run level specified in that entry as the initial run level to enter. If there is no `initdefault` entry in `/etc/inittab`, `init` requests that the user enter a run level from the virtual system console. If an `S` or `s` is entered, `init` goes to the single-user state. In the single-user state the virtual console terminal is assigned to the user's terminal and is opened for reading and writing. The command `/bin/su` is invoked and a message is generated on the physical console saying where the virtual console has been relocated. Use either `init` or `telinit`, to signal `init` to change the run level of the system. Note that if the shell is terminated (via an end-of-file), `init` will only re-initialize to the single-user state if the `/etc/inittab` file does not exist.

If a `0` through `6` is entered, `init` enters the corresponding run level. Note that, on the 3B2 Computer, the run levels `0`, `1`, `5`, and `6` are reserved states for shutting the system down; the run levels `2`, `3`, and `4` are available as normal operating states.

If this is the first time since power up that `init` has entered a run level other than single-user state, `init` first scans `/etc/inittab` for `boot` and `bootwait` entries (see `inittab(4)`). These entries are performed before any other processing of `/etc/inittab` takes place, providing that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. `init` then scans `/etc/inittab` and executes all other entries that are to be processed for that run level.

In a multi-user environment, `/etc/inittab` is set up so that `init` will create a `getty` process for each terminal that the administrator sets up to respawn.

To spawn each process in `/etc/inittab`, `init` reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by `/etc/inittab`, `init` waits for one of its descendant processes to die, a powerfail signal, or a signal from another `init` or `telinit` process to change the system's run level. When one of these conditions occurs, `init` re-examines `/etc/inittab`. New entries can be added to `/etc/inittab` at any time; however, `init` still waits for one of the above three conditions to occur before re-examining `/etc/inittab`. To get around this, `init Q` or `init q` command wakes `init` to re-examine `/etc/inittab` immediately.

When `init` comes up at boot time and whenever the system changes from the single-user state to another run state, `init` sets the `ioctl(2)` states of the virtual console to those modes saved in the file `/etc/ioctl.syscon`. This file is written by `init` whenever the single-user state is entered.

When a run level change request is made *init* sends the warning signal (SIGTERM) to all processes that are undefined in the target run level. *init* waits 5 seconds before forcibly terminating these processes via the kill signal (SIGKILL).

The shell running on each terminal will terminate when the user types an end-of-file or hangs up. When *init* receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in */etc/utmp* and */etc/wtmp* if it exists (see *who(1)*). A history of the processes spawned is kept in */etc/wtmp*.

If *init* receives a *powerfail* signal (SIGPWR) it scans */etc/inittab* for special entries of the type *powerfail* and *powerwait*. These entries are invoked (if the run levels permit) before any further processing takes place. In this way *init* can perform various cleanup and recording functions during the powerdown of the operating system. Note that in the single-user states, S and s, only *powerfail* and *powerwait* entries are executed.

telinit

telinit, which is linked to */etc/init*, is used to direct the actions of *init*. It takes a one-character argument and signals *init* to take the appropriate action.

FILES

/etc/inittab
/etc/utmp
/etc/wtmp
/etc/ioctl.syscon
/dev/console
/dev/contty

SEE ALSO

getty(1M), *shutdown(1M)*, *gettydefs(4)*, *inittab(4)*, *utmp(4)*, *termio(7)*.
login(1), *sh(1)*, *stty(1)*, *who(1)* in the *User's Reference Manual*.
kill(2) in the *Programmer's Reference Manual*.

DIAGNOSTICS

If *init* finds that it is respawning an entry from */etc/inittab* more than 10 times in 2 minutes, it will assume that there is an error in the command string in the entry, and generate an error message on the system console. It will then refuse to respawn this entry until either 5 minutes has elapsed or it receives a signal from a user-spawned *init* (*telinit*). This prevents *init* from eating up system resources when someone makes a typographical error in the *inittab* file or a program is removed that is referenced in */etc/inittab*.

When attempting to boot the system, failure of *init* to prompt for a new run level may be because the virtual system console is linked to a device other than the physical system console.

WARNINGS

init and *telinit* can be run only by someone who is super-user.

The S or s state must not be used indiscriminately in the */etc/inittab* file. A good rule to follow when modifying this file is to avoid adding this state to any line other than the *initdefault*.

The change to `/etc/gettydefs` described in the **WARNINGS** section of the `gettydefs(4)` manual page will permit terminals to pass 8 bits to the system as long as the system is in multi-user state (run level greater than 1). When the system changes to single-user state, the `getty` is killed and the terminal attributes are lost. To permit a terminal to pass 8 bits to the system in single-user state, after you are in single-user state, type:

```
stty -istrip cs8
```

NAME

install – install commands

SYNOPSIS

```
/etc/install [-c dira] [-f dirb] [-i] [-n dirc] [-m mode] [-u user] [-g
group] [-o] [-s] file [dirx ...]
```

DESCRIPTION

The *install* command is most commonly used in “makefiles” [See *make(1)*] to install a *file* (updated target file) in a specific place within a file system. Each *file* is installed by copying it into the appropriate directory, thereby retaining the mode and owner of the original command. The program prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx ...*) are given, *install* will search a set of default directories (*/bin*, */usr/bin*, */etc*, */lib*, and */usr/lib*, in that order) for a file with the same name as *file*. When the first occurrence is found, *install* issues a message saying that it is overwriting that file with *file*, and proceeds to do so. If the file is not found, the program states this and exits without further action.

If one or more directories (*dirx ...*) are specified after *file*, those directories will be searched before the directories specified in the default list.

The meanings of the options are:

- c *dira* Installs a new command (*file*) in the directory specified by *dira*, only if it is not found. If it is found, *install* issues a message saying that the file already exists, and exits without overwriting it. May be used alone or with the –s option.
- f *dirb* Forces *file* to be installed in given directory, whether or not one already exists. If the file being installed does not already exist, the mode and owner of the new file will be set to 755 and *bin*, respectively. If the file already exists, the mode and owner will be that of the already existing file. May be used alone or with the –o or –s options.
- i Ignores default directory list, searching only through the given directories (*dirx ...*). May be used alone or with any other options except –c and –f.
- n *dirc* If *file* is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file will be set to 755 and *bin*, respectively. May be used alone or with any other options except –c and –f.
- m *mode* The mode of the new file is set to *mode*. Only available to the superuser.
- u *user* The owner of the new file is set to *user*. Only available to the superuser.

- g** *group* The group id of the new file is set to *group*. Only available to the superuser.
- o** If *file* is found, this option saves the "found" file by copying it to **OLDfile** in the directory in which it was found. This option is useful when installing a frequently used file such as */bin/sh* or */etc/getty*, where the existing file cannot be removed. May be used alone or with any other options except **-c**.
- s** Suppresses printing of messages other than error messages. May be used alone or with any other options.

SEE ALSO

make(1).

NAME

killall – kill all active processes

SYNOPSIS

/etc/killall [signal]

DESCRIPTION

killall is used by **/etc/shutdown** to kill all active processes not directly related to the shutdown procedure.

killall terminates all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

killall sends *signal* (see *kill*[1]) to all processes not belonging to the above group of exclusions. If no *signal* is specified, a default of 9 is used.

FILES

/etc/shutdown

SEE ALSO

fuser(1M), *shutdown*(1M).

kill(1), *ps*(1) in the *User's Reference Manual*.

signal(2) in the *Programmer's Reference Manual*.

WARNINGS

The *killall* command can be run only by the super-user.



NAME

labelit – provide labels for file systems

SYNOPSIS

```
/etc/labelit special [ fsname volume [ -n ] ]
```

DESCRIPTION

labelit can be used to provide labels for unmounted disk file systems or file systems being copied to tape. The *-n* option provides for initial labeling only (this destroys previous contents).

With the optional arguments omitted, *labelit* prints current label values.

The *special* name should be the physical disk section (e.g., */dev/dsk/c0d0s6*), or the cartridge tape (e.g., */dev/SA/ctape1*). The device may not be on a remote machine.

The *fsname* argument represents the mounted name (e.g., *root*, *u1*, etc.) of the file system.

Volume may be used to equate an internal name to a volume name applied externally to the disk pack, diskette or tape.

For file systems on disk, *fsname* and *volume* are recorded in the superblock.

SEE ALSO

makefsys(1M), *fs*(4).

sh(1) in the *User's Reference Manual*.



NAME

ldsysdump – load system dump from floppy diskettes

SYNOPSIS

/etc/ldsysdump destination_file

DESCRIPTION

The *ldsysdump* command loads the memory image files from the floppy diskettes used to take a system dump and recombines them into a single file on the hard disk suitable for use by *crash(1)*. The *destination_file* is the name of the hard disk file into which the data from the diskettes will be loaded.

When invoked, *ldsysdump* begins an interactive procedure that prompts the user to insert the diskettes to be loaded. The user has the option of quitting the session at any time. This allows only the portion of the system image needed to be dumped.

EXAMPLES

This example loads the 3 floppies produced via *sysdump(8)* on a machine equipped with 2 MB of memory.

```
$ldsysdump /usr/tmp/cdump
```

```
Insert first sysdump floppy.
```

```
Enter 'c' to continue, 'q' to quit: c
```

```
Loading sysdump
```

```
.....
.....
```

```
Insert next sysdump floppy.
```

```
Enter 'c' to continue, 'q' to quit: c
```

```
Loading more sysdump
```

```
.....
.....
```

```
Insert next sysdump floppy.
```

```
Enter 'c' to continue, 'q' to quit: c
```

```
Loading more sysdump
```

```
.....
.....
```

```
3 Sysdump files coalesced, 4096 (512 byte) blocks
$
```

FILES

/dev/dsk/c0d0s6 device used for floppy access

SEE ALSO

crash(1M), *sysdump(8)*.

ulimit(1) in the *User's Reference Manual*.

DIAGNOSTICS

If a floppy diskette is inserted out of sequence a message is printed. The user is allowed to insert a new floppy and continue the session.

WARNINGS

Since the 3B2 computer can be equipped with up to 4 MB of memory, the *destination_file* can become quite large. The file size limit must be set large enough to hold a file of this size.

NAME

led - flash green LED

SYNOPSIS

/etc/led [-f] [-o]

DESCRIPTION

led is used to turn on the green LED (light emitting diode) located on the exterior of the cabinet. The main purpose is to signal particular phases of the boot procedure. The options are as follows:

- f sets the green LED to a flashing state via the *sys3b(2)* system call.
- o sets the green LED to a constant on state via the *sys3b(2)* system call.

SEE ALSO

sys3b(2) in the *Programmer's Reference Manual*.

WARNINGS

This command can be run only by the super-user.



NAME

link, unlink – link and unlink files and directories

SYNOPSIS

```
/etc/link file1 file2
/etc/unlink file
```

DESCRIPTION

The *link* command is used to create a file name that points to another file. Linked files and directories can be removed by the *unlink* command; however, it is strongly recommended that the *rm(1)* and *rmdir(1)* commands be used instead of the *unlink* command.

The only difference between *ln(1)* and *link/unlink* is that the latter do exactly what they are told to do, abandoning all error checking. This is because they directly invoke the *link(2)* and *unlink(2)* system calls.

SEE ALSO

rm(1) in the *User's Reference Manual*.
link(2), *unlink(2)* in the *Programmer's Reference Manual*.

WARNINGS

These commands can be run only by the super-user.



NAME

`lpadmin` – configure the LP print service

SYNOPSIS

```

/usr/lib/lpadmin -p printer options
/usr/lib/lpadmin -x dest
/usr/lib/lpadmin -d [dest]
/usr/lib/lpadmin -S print-wheel -A alert-type [-W integer1] [-Q integer2]

```

DESCRIPTION

`lpadmin` configures the LP print service by defining printers and devices. It is used to add and change printers, to remove printers from the service, to set or change the system default destination, and to define alerts for print wheels.

Adding or Changing a Printer

The first form of the `lpadmin` command (`lpadmin -p printer options`) is used to configure a new printer or to change the configuration of an existing printer. The following *options* are used, and may appear in any order. For ease of discussion, the printer will be referred to as *P* below.

-F *fault-recovery*

Restore the LP print service after a printer fault, according to the value of *fault-recovery*:

continue

Continue printing on the top of the page where printing stopped. This requires a filter to wait for the fault to clear before automatically continuing.

beginning

Start printing the request again from the beginning.

wait Disable printing on the printer and wait for the administrator or a user to enable printing again.

During the wait the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. If no change request is made before printing is enabled, printing will resume at the top of the page where stopped, if the filter allows; otherwise, the request will be printed from the beginning.

This option specifies the recovery to be used for any print request that is stopped because of a printer fault.

-c *class*

Insert printer *P* into the specified *class*. *Class* will be created if it does not already exist.

-D *comment*

Save this *comment* for display whenever a user asks for a full description of the printer *P* (see `lpstat(1)`). The LP print service does not interpret this comment.

-e printer

Copy an existing *printer's* interface program to be the interface program for printer *P*.

-f allow:form-list**-f deny:form-list**

Allows (**-f allow**) or denies (**-f deny**) the forms in *form-list* to be printed on printer *P*.

For each printer, the LP print service keeps two lists of forms: an "allow-list" of forms that can be used with the printer, and a "deny-list" of forms that shouldn't be used with the printer. With the **-f allow** option, the forms listed are added to the allow-list and removed from the deny-list. With the **-f deny** option, the forms listed are removed from the allow-list and added to the deny-list.

If the allow-list is not empty, the forms in the list can be used with the printer and all others cannot, regardless of the content of the deny-list. If the allow-list is empty, but the deny-list is not, the forms in the deny-list cannot be used with the printer. All forms can be excluded from a printer by having an empty allow-list and putting the word **any** in the deny-list. All forms can be used on a printer by having an empty deny-list and specifying **any** for the allow-list, provided the printer can handle all the characteristics of the forms.

The LP print service uses this information as a set of guidelines for determining where a form can be mounted. Administrators, however, are not restricted from mounting a form on any printer. If mounting a form on a particular printer is in disagreement with the information in the allow-list or deny-list, the administrator is warned but the mount is accepted. Nonetheless, if a user attempts to issue a print or change request for a form and printer combination that is in disagreement with the information, the request is accepted only if the form is currently mounted on the printer. If the form is later unmounted before the request can print, the request is canceled and the user notified by mail.

If an administrator tries to name a form as acceptable for use on a printer that doesn't have the capabilities needed by the form, the command is rejected.

Note the other use of **-f** below.

-h Indicates that the device associated with *P* is hardwired. This option is assumed when adding a new printer unless the **-l** option is supplied.**-i interface**

Establishes a new interface program for *P*. *Interface* is the path name of the new program.

-I content-type-list

Assigns *P* to handle print requests with content of a type listed in *content-type-list*.

The type **simple** is recognized as the default content-type of files in the UNIX system. Such a data stream contains only printable ASCII characters and the following control characters.

Control Character	Octal Value	Meaning
backspace	10 ₈	move back to previous column, except at beginning of line
tab	11 ₈	move to next tab stop
linefeed (newline)	12 ₈	move to beginning of next line
form feed	14 ₈	move to beginning of next page
carriage return	15 ₈	move to beginning of current line

To force the print service to not consider **simple** as a valid type for the printer, give an explicit value (e.g., the printer type) in the *content-type-list*. Vice versa, if you do want **simple** included along with other types, you must include **simple** in the *content-type-list*.

Each printer automatically has its printer type included in the list of content types it will accept.

Except for **simple**, each *content-type* name is freely determined by the administrator. If names given as content types are also printer types, the names are accepted without comment, because the LP print service recognizes all printer types as potential content types as well.

-I Indicates that the device associated with *P* is a login terminal. The LP scheduler, *lpsched*, disables all login terminals automatically each time it is started. Before re-enabling *P*, its current *device* should be established using *lpadmin*.

-M -f *form-name* [-a [-o **filebreak**]]

Mounts the form *form-name* on *P*. Print requests to be printed with the pre-printed form *form-name* will be printed on *P*. If more than one printer has the form mounted and the user has specified any (with the -d option of the **lp** command) as the printer destination, then the print request will be printed on the one that also meets the other needs of the request.

The page length and width, and character and line pitches needed by the form are compared with those allowed for the printer, by checking the capabilities in the *terminfo*(4) database for the type of printer. If the form requires attributes that are not available with the printer, the administrator is warned but the mount is accepted. If the form lists a print wheel as mandatory, but the print wheel mounted on the printer is different, the administrator is also warned but the mount is accepted.

If the -a option is given, an alignment pattern is printed, preceded by the same initialization of the physical printer that precedes a normal print request, with one exception: no banner page is printed. Printing is assumed to start at the top of the first page of the form. After the pattern is printed, the administrator can adjust the mounted form in the

printer and press return for another alignment pattern (no initialization this time), and can continue printing as many alignment patterns as desired. The administrator can quit the printing alignment patterns by typing 'q'.

If the **-o filebreak** option is given, a formfeed is inserted between each copy of the alignment pattern. By default, the alignment pattern is assumed to correctly fill a form, so no formfeed is added.

A form is "unmounted" by mounting a new form in its place using the **-f** option or the **-f none** option. By default, a new printer has no form mounted.

Note the other use of **-f** above.

-M -S print-wheel

Mounts the print wheel *print-wheel* on *P*. Print requests to be printed with *print-wheel* will be printed on *P*. If more than one printer has *print-wheel* mounted and the user has specified any (with the **-d** option of the **lp** command) as the printer destination, then the print request will be printed on the one that also meets the other needs of the request.

If the *print-wheel* is not listed as acceptable for the printer, the administrator is warned but the mount is accepted. If the printer does not take print wheels, the command is rejected.

A print wheel is "unmounted" by mounting a new print wheel in its place or by using the option **-S none**.

By default, a new printer has no special print wheel mounted. Until this is changed, a print request that asks for a specific print wheel will not be printed on *P*.

Note the other uses of the **-S** option described below.

-m model

Selects a model interface program, provided with the LP print service, for printer *P*.

-o printing-option

Each **-o** option in the list below is the default given to an interface program if the option is not taken from a preprinted form description or is not explicitly given by the user submitting a request (see *lp(1)*). The only **-o** options that can have defaults defined are listed below.

```
length=scaled-decimal-number
width=scaled-decimal-number
cpi=scaled-decimal-number
lpi=scaled-decimal-number
stty='stty-option-list'
```

The term "scaled-decimal-number" refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a "trailing" letter attached to the number. Three types of scaled decimal numbers can be used with the LP print service: numbers that show sizes in centimeters

(marked with a trailing "c"); numbers that show sizes in inches (marked with a trailing "i"); and numbers that show sizes in units appropriate to use (without a trailing letter), i.e., lines, columns, lines per inch, or characters per inch.

The first four default option values should agree with the capabilities of the type of physical printer, as defined in the *terminfo(4)* database for the printer type. If they do not, the command is rejected.

The *stty-option-list* is not checked for allowed values, but is passed directly to the *stty(1)* program by the standard interface program. Any error messages produced by *stty(1)* when a request is processed (by the standard interface program) are mailed to the user submitting the request.

For each printing option not specified, the defaults for the following attributes are defined in the Terminfo entry for the specified printer type.

length
width
cpi
lpi

The default for *stty* is

```
stty=9600 cs8 -cstopb -parenb -paroff ixon
      -ixany opost -olcuc -onlcr -ocrnl -onocr
      -onlret -ofill nl0 cr0 tab0 bs0 vt0 ff0
```

You can set any of the *-o* options to the default values (which vary for different types of printers), by typing them without assigned values, as follows:

length=
width=
cpi=
lpi=
stty=

-o nobanner

Allows users to submit a print request that asks that no banner page be printed.

-o banner

Forces a banner page to be printed with every print request, even when a user asks for no banner page. This is the default; you must specify *-o nobanner* if you want to allow users to specify *-o nobanner* with the *lp* command.

-r class

Removes printer *P* from the specified *class*. If *P* is the last member of the printer class *class*, then *class* will be removed.

-S list

Allows the print wheels or aliases for character sets named in *list* to be used with *P*.

If the printer is a type that takes print wheels, then *list* is a comma or space separated list of print wheel names. (Enclose the list with quotes if it contains blanks.) These will be the only print wheels considered mountable on the printer. (You can always force a different print wheel to be mounted, however.) Until the option is used to specify a list, no print wheels will be considered mountable on the printer, and print requests that ask for a particular print wheel with this printer will be rejected.

If the printer is a type that has selectable character sets, then *list* is a comma or separated list of character set name "mappings" or aliases. (Enclose the list with quotes if it contains blanks.) Each "mapping" is of the form

known-name=alias

The *known-name* is: a character set number preceded by **cs**, such as **cs3** for character set three; or a character set name from the Terminfo database **csnm** entry. (See *terminfo(4)* in the *Programmer's Reference Manual*.) If this option is not used to specify a list, only the names already known from the Terminfo database or numbers with a prefix of **cs** will be acceptable for the printer.

If *list* is the word **none**, any existing print wheel list or character set aliases will be removed.

Note the other uses of the **-S** option

-T printer-type

Assigns the given *printer-type*, a representation of a physical printer of type *printer-type*. *Printer-type* is used to extract data from *terminfo(4)*; this data is used to initialize the printer before printing each user's request. Some filters may also use *printer-type* to convert content for the printer. If this option is not used, the default *printer-type* will be **unknown**; no useful information will be extracted from *terminfo(4)* so each user request will be printed without first initializing the printer. Also, this option must be used if the following are to work: **-o cpi=**, **-o lpi=**, **-o width=**, and **-o length=** options of the **lpadmin** and **lp** commands, and the **-S** and **-f** options of the **lpadmin** command.

-u allow:user-list**-u deny:user-list**

Allows (**-u allow**) or denies (**-u deny**) the users in *user-list* access to *P*.

For normal access to each printer the LP print service keeps two lists of users: an "allow-list" of people allowed to use the printer, and a "deny-list" of people denied access to the printer. With the **-u allow** option,

the users listed are added to the allow-list and removed from the deny-list. With the **-u deny** option, the users listed are removed from the allow-list and added to the deny-list.

If the allow-list is not empty, the users in the list are allowed access to the printer and all others are denied access, regardless of the content of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list are denied access and all others are allowed. If both lists are empty, all users are allowed access. Access can be denied to all users, except the LP print service administrator, by putting **any** in the deny-list. To allow everyone access to *P* and effectively empty both lists, put **any** in the allow-list.

-U dial-info

Assign the "dialing" information *dial-info* to the printer. *Dial-info* is used with the *dial(3)* routine to call the printer. Any network connection supported by the Basic Networking Utilities will work. *Dial-info* can be either a phone number for a modem connection, or a system name for other kinds of connections. Or, if **-U direct** is given, no dialing will take place, because the name **direct** is reserved for a printer that is directly connected. If a system name is given, it is used to search for connection details from the file */usr/lib/uucp/Systems* or related files. The Basic Networking Utilities are required to support this option. By default, **-U direct** is assumed.

-v device

Associates a *device* with printer *P*. *Device* is the path name of a file that is writable by *lp*. Note that the same *device* can be associated with more than one printer.

-A alert-type [-W integer]

The **-A** option is used to define an alert-type to inform the administrator when a printer fault is detected, and periodically thereafter, until the printer fault is cleared by the administrator. The *alert-types* are:

- mail** Send the alert message via mail (see *mail(1)*) to the administrator who issues this command.
- write** Write the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.
- quiet** Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the fault has been cleared and printing resumes, messages will again be sent when another fault occurs with the printer.
- none** Do not send messages; any existing alert definition for the printer will be removed. No alert will be sent when the printer faults until a different alert-type (except **quiet**) is used.

shell-command

The *shell-command* is run each time the alert needs to be sent. The shell command should expect the message as standard

input. If there are blanks embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail** *user-name* and **write** *user-name* respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command *unless* he or she has used the *su(1)* command to change to another user ID. If the *su(1)* command has been used to change the user ID, then the *user-name* for the new ID is used.

- list** The type of the alert for the printer fault is displayed on the standard output. No change is made to the alert.

The message sent appears as follows:

```
The printer printer-name has stopped printing for the reason
given below, Fix the problem and bring the printer back on
line. Printing has stopped, but will be restarted in a few
minutes; issue an enable command if you want to restart
sooner. Unless someone issues a change request
```

```
lp -i request-id -P ...
```

```
to change the page list to print, the current request will
be reprinted from the beginning.
```

```
The reason(s) it stopped (multiple reasons indicate
reprinted attempts):
```

```
reason
```

The LP print service can detect printer faults only through an adequate fast filter and only when the standard interface program or a suitable customized interface program is used. Furthermore, the level of recovery after a fault depends on the capabilities of the filter.

If the *printer-name* is **all**, the alerting defined in this command applies to all existing printers.

If the **-W** option is not used to arrange fault alerting for a printer, the default procedure is to mail one message to the administrator of the printer per fault. Similarly, if *integer* is zero, only one message will be sent per fault. If *integer* is a non-zero number, an alert will be sent every *integer* minute(s).

Restrictions

When creating a new printer, either the **-v** or the **-U** option must be supplied. In addition, only one of the following may be supplied: **-e**, **-i**, or **-m**; if none of these three options is supplied, the model standard is used. The **-h** and **-l** keyletters are mutually exclusive. Printer and class names may be no longer than 14 characters and must consist entirely of the characters **A-Z**, **a-z**, **0-9** and **_** (underscore).

Removing a Printer Destination

The `-x dest` option removes the destination *dest* from the LP print service. If *dest* is a printer and is the only member of a class, then the class will be deleted, too. If *dest* is **all**, all printers and classes are removed. No other *options* are allowed with `-x`.

Setting/Changing the System Default Destination

The `-d [dest]` option makes *dest*, an existing destination, the new system default destination. If *dest* is not supplied, then there is no system default destination. No other *options* are allowed with `-d`.

Setting an Alert for a Print Wheel

`-S print-wheel -A alert-type [-W integer1] [-Q integer2]`

The `-S print-wheel` option is used with the `-A alert-type` option to send the alert *alert-type* to the administrator to mount the print wheel when there is one or more jobs queued for it. If this command is not used to arrange alerting for a print wheel, no alert will be sent for the print wheel. The *alert-types* are:

- mail** Send the alert message via mail (see *mail(1)*) to the administrator who issues this command.
- write** Write the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.
- quiet** Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the *print-wheel* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests again exceeds the threshold.
- none** Do not send messages until the `-A` option is given again with a different *alert-type* (other than **quiet**).

shell-command

The *shell-command* is run each time the alert needs to be sent. The shell command should expect the message as standard input. If there are blanks embedded in the command, enclose the command in quotes. Note that the **mail** and **write** values for this option are equivalent to the values **mail user-name** and **write user-name** respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command *unless* he or she has used the *su(1)* command to change to another user ID. If the *su(1)* command has been used to change the user ID, then the *user-name* for the new ID is used.

- list** The type of the alert for the print wheel is displayed on the standard output. No change is made to the alert.

The message sent appears as follows:

```
The print wheel print-wheel needs to be mounted
on the printer(s):
printer (integer3 requests)
integer4 print requests await this print wheel.
```

The printers listed are those that the administrator had earlier specified were candidates for this print wheel. The number (*integer*₃) listed next to each printer is the number of requests eligible for the printer. The number (*integer*₄) shown after the printer list is the total number of requests awaiting the print wheel. It will be less than the sum of the other numbers if some requests can be handled by more than one printer.

If the *print-wheel* is **all**, the alerting defined in this command applies to all print wheels already defined to have an alert.

If the **-W** option is not given or *integer*₁ is zero (which is interpreted as **once** and is also the default), only one message will be sent per need to mount a print wheel. If *integer*₁ is a non-zero number, an alert will be sent every *integer*₁ minute(s).

If the **-Q** option is also given, the alert will be sent when *integer*₂ print requests that need the print wheel are waiting. If the **-Q** option is not given, or *integer*₂ is 1 or the word **any** (which are both the default), a message is sent as soon as anyone submits a print request for the print wheel when it is not mounted.

The **-S** option has a different meaning when used with the **-p** option.

FILES

/usr/spool/lp/*

SEE ALSO

accept(1M), lpsched(1M).
 enable(1), lp(1), lpstat(1), stty(1) in the *User's Reference Manual*.
 dial(3), terminfo(4) in the *Programmer's Reference Manual*.

NAME

`lpfilter` – administer filters used with the LP print service

SYNOPSIS

```

/usr/lib/lpfilter -f filter-name -F path-name
/usr/lib/lpfilter -f filter-name -
/usr/lib/lpfilter -f filter-name -i
/usr/lib/lpfilter -f filter-name -x
/usr/lib/lpfilter -f filter-name -l

```

DESCRIPTION

The `lpfilter` command is used to add, change, delete, and list filters used with the LP print service. These filters are used to convert the content type of a file to a content type acceptable to a given printer. One of the following options must be used with the `lpfilter` command: `-F path-name` (or `-` for standard input) to add or change a filter; `-i` to reset an original LP print service filter to its factory setting; `-x` to delete a filter; or `-l` to list a filter description.

The argument `all` can be used instead of a `filter-name` with any of these options. When `all` is specified with the `-F` or `-` option, the requested change is made to all filters. Using `all` with the `-i` option has the effect of restoring to their original settings all filters for which predefined settings were initially available. Using the `all` argument with the `-l` option produces a list of all filters, and using it with the `-x` option results in all filters being deleted.

Adding or Changing a Filter

The filter named in the `-f` option and described in the input is added to the filter table. If the filter already exists, its description is changed to reflect the new information in the input. Once added, a filter is available for use.

The filter description is taken from the `path-name` if the `-F` option is given, or from the standard input if the `-` option is given. One of the two must be given to define or change a filter. If the filter named is one originally delivered with the LP print service, the `-i` option will restore the original filter description.

When an existing filter is changed with the `-F` or `-` option, items that are not specified in the new information are left as they were. When a new filter is added with this command, unspecified items are given default values.

Note that a filter name and a command must be given. A filter with no input type value is assumed to work with any input type; this is also true for the output type, printer type, and printer values.

Filters are used to convert the content of a request into a data stream acceptable to a printer. For a given print request, the LP print service will know the following: the type of content in the request, the name of the printer, the type of the printer, the types of content acceptable to the printer, and the modes of printing asked for by the originator of the request. It will use this information to find a filter that will convert the content into a type acceptable to the printer.

Below is a list of items that provide input to this command, and a description of each item. All lists are comma or space separated.

Input types: *content-type-list*
Output types: *content-type-list*
Printer types: *printer-type-list*
Printers: *printer-list*
Filter type: *filter-type*
Command: *shell-command*
Options: *template-list*

Input types

This gives the types of content that can be accepted by the filter.

Output types

This gives the types of content that the filter can produce from any of the input content types.

Printer types

This gives the type of printers for which the filter can be used. The LP print service will restrict the use of the filter to these types of printers.

Printers

This gives the names of the printers for which the filter can be used. The LP print service will restrict the use of the filter to just the printers named.

Filter type

This marks the filter as a "slow" filter or a "fast" filter. Slow filters are generally those that take a long time to convert their input. They are run unconnected to a printer, to keep the printers from being tied up while the filter is running. Fast filters are generally those that convert their input quickly, or those that must be connected to the printer when run. These will be given to the interface program to run connected to the physical printer.

Command

This specifies the program to run to invoke the filter. The program name as well as fixed options are included in the *shell-command*; additional options are constructed, based on the characteristics of each print request and on the **Options** field.

The command must accept a data stream as standard input and produce the converted data stream on its standard output. This allows filter pipelines to be constructed to convert data not handled by a single filter.

Options

This is a comma separated list of templates used by the LP print service to construct options to the filter from the characteristics of each print request listed in the table later.

In general, each template is of the following form:

keyword pattern = replacement

The *keyword* names the characteristic that the template attempts to

map into a filter specific option; each valid *keyword* is listed in the table below. A *pattern* is either a literal pattern of one of the forms listed in the table, or a single asterisk (*); if the *pattern* matches the value of the characteristic, the template fits and is used to generate a filter specific option. A *pattern* of * matches any value. The *replacement* is a string used as a filter specific option; if an asterisk (*) is used, it is replaced with the value of the characteristic.

lp Option	Characteristic	<i>keyword</i>	Possible <i>patterns</i>
-T	Content type (input)	INPUT	<i>content-type</i>
N/A	Content type (output)	OUTPUT	<i>content-type</i>
N/A	Printer type	TERM	<i>printer-type</i>
-f, -o cpi=	Character pitch	CPI	<i>integer</i>
-f, -o lpi=	Line pitch	LPI	<i>integer</i>
-f, -o length=	Page length	LENGTH	<i>integer</i>
-f, -o width=	Page width	WIDTH	<i>integer</i>
-P	Pages to print	PAGES	<i>page-list</i>
-S	Character set/ print wheel	CHARSET	<i>character-set-name/ print-wheel-name</i>
-f	Form name	FORM	<i>form-name</i>
-y	Modes	MODES	<i>mode</i>
-n	Number of copies	COPIES	<i>integer</i>

For example, the template

MODES landscape = -l

would show that if a print request was submitted with the **-y landscape** option, the filter should be given the option **-l**. As another example, the template

TERM * = -T *

would show that the filter should be given the option **-T printer-type** for whichever *printer-type* is associated with a print request using the filter.

Deleting a Filter

The **-x** option is used to delete the filter specified in *filter-name* from the LP filter table.

Listing a Filter Description

The **-l** option is used to list the description of the filter named in *filter-name*. If the command is successful, the following message is sent to standard output:

```
Input types: content-type-list
Output types: content-type-list
Printer types: printer-type-list
Printers: printer-list
Filter type: filter-type
```

LPFILTER(1M)

LPFILTER(1M)

Command: *shell-command*
Options: *template-list*

If the command fails, an error message is sent to standard error.

SEE ALSO

lpadm(1M).
lp(1) in the *User's Reference Manual*

NAME

lpforms – administer forms used with the LP print service

SYNOPSIS

```

/usr/lib/lpforms -f form-name options
/usr/lib/lpforms -f form-name -A alert-type [-Q integer1] [-W integer2]
/usr/lib/lpforms -f form-name -A list
/usr/lib/lpforms -f form-name -A quiet
/usr/lib/lpforms -f form-name -A none

```

DESCRIPTION

The *lpforms* command is used to administer the use of preprinted forms, such as company letterhead paper, with the LP print service. A form is specified by the *form-name* given with the *lpforms* command. Users may request a form by *form-name* (see *lp(1)*). The argument *all* can be used instead of *form-name* with any of the five *lpforms* command lines shown above. The first command line allows the administrator to add, change, and delete forms, to list the attributes of an existing form, and to allow and deny users access to particular forms. The second command line is used to establish the method by which the administrator is alerted that a form must be mounted on a printer. By using the third command line, an administrator can list the current alerting methods assigned to forms. The fourth command *lpforms* command line is used to terminate an active alert, and the fifth command line is used to remove an alert definition.

With the first *lpforms* command line, one of the following options must be used:

-F <i>path-name</i>	to add or change a form as specified by the information in <i>path-name</i>
-	to add or change a form, and supply information from standard input
-x	to delete a form (this option must be used separately; it cannot be used with any other option)
-l	to list the attributes of a form
-u allow: <i>user-list</i>	to allow users to request a form (this option can be used with the -F or - option)
-u deny: <i>user-list</i>	to deny users access to a form (this option can be used with the -F or - option)

Each option is explained below.

Adding or Changing a Form

The **-F** *path-name* option is used to add a new form to the LP print service, or to change the attributes of an existing form. The form description is taken from *path-name* if the **-F** option is given, or the standard input if the **-** option is used. One of the two options must be used to define or change a form. *path-name* is the path name of a file that contains all or any subset of the following information about the form.

Page length: *scaled—decimal—number₁*
Page width: *scaled—decimal—number₂*
Number of pages: *integer*
Line pitch: *scaled—decimal—number₃*
Character pitch: *scaled—decimal—number₄*
Character set choice: *character-set/print-wheel,[mandatory]*
Ribbon color: *ribbon-color*
Comment:
comment
Alignment pattern: *[content-type]*
content

Except for the last two lines, the above lines can appear in any order. The **Comment:** and *comment* items must appear in consecutive order but can appear before the other items, and the **Alignment pattern:** and the *content* items must appear in consecutive order at the end of the file. Also, the *comment* item cannot contain a line that begins with any of the key phrases above, unless the key phrase is preceded with a > sign. Any leading > sign found in the *comment* will be removed when the comment is displayed. Case distinctions in the key phrases are ignored.

When this command is issued, the form specified by *form-name* is added to the list of forms. If the form already exists, its description is changed to reflect the new information in the input. Once added, a form is available for use in a print request, except where access to the form has been restricted, as described under the **-u allow:** option. A form may also be allowed to be used on certain printers only.

A description of each form attribute is below:

Page length and Page Width

Before printing the content of a print request needing this form, the generic interface program provided with the LP print service will initialize the physical printer to handle pages *scaled—decimal—number₁* long, and *scaled—decimal—number₂* wide using the printer type as a key into the *terminfo(4)* database. A *scaled-decimal-number* is an optionally scaled decimal number that gives a size in lines, columns, inches, or centimeters, as appropriate. The scale is indicated by appending the letter "i" for inches, or the letter "c" for centimeters. For length or width settings, an unscaled number indicates lines or columns; for line pitch or character pitch settings, an unscaled number indicates lines per inch or characters per inch (the same as a number scaled with "i"). For example, **length=66** indicates a page length of 66 lines, **length=11i** indicates a page length of 11 inches, and **length=27.94c** indicates a page length of 27.94 centimeters.

The page length and page width will also be passed, if possible, to each filter used in a request needing this form.

Number of pages

Each time the alignment pattern is printed, the LP print service will

attempt to truncate the *content* to a single form by, if possible, passing to each filter the page subset of 1-*integer*.

Line pitch and Character pitch

Before printing the content of a print request needing this form, the interface programs provided with the LP print service will initialize the physical printer to handle these pitches, using the printer type as a key into the *terminfo*(4) database. Also, the pitches will be passed, if possible, to each filter used in a request needing this form. *Scaled-decimal-number*₃ is in lines per centimeter if a "c" is appended, and lines per inch otherwise; similarly, *scaled-decimal-number*₄ is in columns per centimeter if a "c" is appended, and columns per inch otherwise. The character pitch can also be given as **elite** (12 characters per inch), **pica** (10 characters per inch), or **compressed** (as many characters per inch as possible).

Character set choice

When the LP print service alerts an administrator to mount this form, it will also mention that the print wheel *print-wheel* should be used on those printers that take print wheels. If printing with this form is to be done on a printer that has selectable or loadable character sets instead of print wheels, the interface programs provided with the LP print service will automatically select or load the correct character set. If **mandatory** is appended, a user is not allowed to select a different character set for use with the form; otherwise, the character set or print wheel named is a suggestion and a default only.

Ribbon color

When the LP print service alerts an administrator to mount this form, it will also mention that the color of the ribbon should be *ribbon-color*.

Comment

The LP print service will display the *comment* unaltered when a user asks about this form (see *lpstat*(1)).

Alignment pattern

When mounting this form an administrator can ask for the *content* to be printed repeatedly, as an aid in correctly positioning the pre-printed form. The optional *content-type* defines the type of printer for which *content* had been generated. If *content-type* is not given, **simple** is assumed. Note that the *content* is stored as given, and will be readable only by the user *lp*.

When an existing form is changed with this command, items missing in the new information are left as they were. When a new form is added with this command, missing items will get the following defaults:

Page Length: 66
Page Width: 80
Number of Pages: 1
Line Pitch: 6

Character Pitch: 10
 Character Set Choice: **any**
 Ribbon Color: **any**
 Comment: (no default)
 Alignment Pattern: (no default)

Deleting a Form

The **-x** option is used to delete the form specified in *form-name* from the LP print service.

Listing Form Attributes

The **-l** option is used to list the attributes of the existing form specified by *form-name*. The attributes listed are those described under "Adding and Changing a Form," above. Because of the potentially sensitive nature of the alignment pattern, only the administrator can examine the form with this command. Other people can use the *lpstat(1)* command to examine the non-sensitive part of the form description.

Allowing and Denying Access to a Form

The LP print service keeps two lists of users for each form: an "allow-list" of people allowed to use the form, and a "deny-list" of people denied access to the form. With the **-u allow:** option, the users listed are added to the allow-list and removed from the deny-list. With the **-u deny:** option, the users listed are removed from the allow-list and added to the deny-list.

If the allow-list is not empty, the users in the list are allowed access to the form and all others are denied access, regardless of the content of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list are denied access and all others are allowed. If both lists are empty, all users are allowed access. Access can be denied to all users, except the LP print service administrator, by putting **any** in the deny-list. To effectively empty both lists, allowing access for everyone, put **any** in the allow-list.

Alerting to Mount Forms

The second *lpforms* command line (shown under "Synopsis") is used to arrange for alerts to be sent to the administrator when forms need to be mounted on a printer.

When *integer*₁ print requests needing the preprinted form *form-name* become queued up because no printer satisfying all the needs of the requests has the form mounted (and for as long as this condition remains), an alert is sent to the administrator every *integer*₂ minutes until the form is mounted on a qualifying printer. If the *form-name* is **all**, the alerting defined in this command applies to all existing forms. No alerting is done for a backlog of print requests needing a form if the administrator does not use this command.

The method by which the alert is sent depends on the value of the **-A** option.

write The message is sent via *write(1)* to the terminal on which the administrator is logged in when the alert arises. If the administrator is logged in on several terminals, one is arbitrarily chosen.

mail The message is sent via *mail(1)* to the administrator who issues this command.

The message sent appears as follows:

```
The form form-name needs to be mounted on the
printer(s) printer (integer5 requests).
integer4 print requests await this form.
Use the ribbon-color ribbon.
Use the print-wheel print wheel, if appropriate.
```

The printers listed are those that the administrator had earlier specified were candidates for this form. The number (*integer*₃) listed next to each printer is the number of requests eligible for the printer. The number (*integer*₄) shown after the list of printers is the total number of requests awaiting the form. It will be less than the sum of the other numbers if some requests can be handled by more than one printer. The *ribbon-color* and *print-wheel* are those specified in the form description. The last line in the message is always sent, even if none of the printers listed use print wheels, because the administrator may choose to mount the form on a printer that does use a print wheel.

Where any color ribbon or any print wheel can be used, the statements above will read:

```
Use any ribbon.
Use any print-wheel.
```

shell-command

The *shell-command* is run each time the alert needs to be sent. The shell command should expect the message as standard input. Note that the **mail** and **write** values for the **-A** command are equivalent to the values "**mail user-name**" and "**write user-name**," respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command *unless* he or she has used the *su* command to change to another user ID. If the *su* command has been used to change the user ID, then the *user-name* for the new ID is used.

If the **-Q** option is not given or *integer*₁ is one or the word **any** (which are both the default), a message is sent as soon as anyone submits a print request for the form when it is not mounted.

If the **-W** option is not given or *integer*₂ is zero or the word **once** (which are both the default), only one message is sent when the queue size exceeds *integer*₁. If *integer*₂ is a non-zero number, an alert will be sent every *integer*₂ minute(s).

Listing the Current Alert

The third *lpforms* command line (shown under "Synopsis") is used to list the type of the alert for the specified form. No change is made to the alert. If *form-name* is recognized by the LP print service, one of the following lines is sent to the standard output, depending on the type of alert for the form.

- When *integer*₁ are queued: alert with "*shell-command*" every *integer*₂ minutes

- When *integer*₁ are queued: write to *user-name* every *integer*₂ minutes
- When *integer*₁ are queued: mail to *user-name* every *integer*₂ minutes
- No alert

The phrase every *integer* minutes is replaced with once if *integer*₂ (*-W integer*₂) is 0.

Terminating an Active Alert

The *-A quiet* option is used to stop messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form has been mounted and then unmounted, messages will again be sent when the queue size reaches *integer*₁ pending requests.

Removing an Alert Definition

No messages will be sent after the *-A none* option is used until the *-A* option is given again with a different *alert-type*. This can be used to permanently stop further messages from being sent as any existing alert definition for the form will be removed.

SEE ALSO

lpadmin(1M), *terminfo*(4).
lp(1) in the *User's Reference Manual*.

NAME

lpsched, *lpshut*, *lpmove* – start/stop the LP print service and move requests

SYNOPSIS

```
/usr/lib/lpsched  
/usr/lib/lpshut  
/usr/lib/lpmove requests dest  
/usr/lib/lpmove dest1 dest2
```

DESCRIPTION

lpsched starts the LP print service; this can be done only by **root** or **lp**.

lpshut shuts down the print service. All printers that are printing at the time *lpshut* is invoked will stop printing. When *lpsched* is started again, requests that were printing at the time a printer was shut down will be reprinted from the beginning.

lpmove moves requests that were queued by *lp*(1) between LP destinations. The first form of the *lpmove* command shown above (under "Synopsis") moves the named *requests* to the LP destination *dest*. *Requests* are request-ids as returned by *lp*(1). The second form of the *lpmove* command moves all requests for destination *dest₁* to destination *dest₂*; *lp*(1) will then reject any new requests for *dest₁*.

Note that when moving requests, *lpmove* never checks the acceptance status (see *accept*(1M)) of the new destination. Also, the request-ids of the moved request are not changed, so that users can still find their requests. The *lpmove* command will not move requests that have options (content type, form required, and so on) that cannot be handled by the new destination.

If a request was originally queued for a class or the special destination **any**, its destination will be changed to *new-destination*. A request thus affected will be printable only on *new-destination* and not on other members of the **class** or other acceptable printers if the original destination was **any**.

NOTE

By default, the directory */usr/spool/lp* is used to hold all the files used by the LP print service. This can be changed by setting the **SPOOLDIR** environment variable to another directory before running *lpsched*. If you do this, you should populate the directory with the same files and directories found under */usr/spool/lp*; the LP print service will not automatically create them. Also, the **SPOOLDIR** variable must then be set before any of the other LP print service commands are run.

FILES

*/usr/spool/lp/**

SEE ALSO

accept(1M), *lpadmin*(1M),
enable(1), *lp*(1), *lpstat*(1) in the *User's Reference Manual*.



NAME

lpusers – set printing queue priorities

SYNOPSIS

```

/usr/lib/lpusers -d priority-level
/usr/lib/lpusers -q priority-level -u user-list
/usr/lib/lpusers -u user-list
/usr/lib/lpusers -q priority-level
/usr/lib/lpusers -l

```

DESCRIPTION

The *lpusers* command is used to set limits to the queue priority level that can be assigned to jobs submitted by users of the LP print service.

The first form of the command (with *-d*) sets the system-wide priority default to *priority-level*, where *priority-level* is a value of 0 to 39, with 0 being the highest priority. If a user does not specify a priority level with a print request (see *lp(1)*), the default priority is used. Initially, the default priority level is 20.

The second form of the command (with *-q* and *-u*) sets the default highest *priority-level* (0-39) that the users in *user-list* can request when submitting a print request. Users that have been given a limit cannot submit a print request with a higher priority level than the one assigned, nor can they change a request already submitted to have a higher priority. Any print requests with priority levels higher than allowed will be given the highest priority allowed.

The third form of the command (with *-u*) removes the users from any explicit priority level, and returns them to the default priority level.

The fourth form of the command (with *-q*) sets the default highest priority level for all users not explicitly covered by the use of the second form of this command.

The last form of the command (with *-l*) lists the default priority level and the priority limits assigned to users.

SEE ALSO

lp(1) in the *User's Reference Manual*.



NAME

makefsys – create a file system on a diskette

SYNOPSIS

makefsys

DESCRIPTION

This command allows the user to create a file system on a diskette. It also writes an internal label in the file system super-block.

The user is asked some questions before the file system is created. Once created, the diskette is self-identifying.

The identical function is available under the *sysadm* menu:

sysadm makefsys

The command may be assigned a password. See *sysadm(1)*, the **admpasswd** sub-command.

SEE ALSO

checkfsys(1M), *labelit(1M)*, *mkfs(1M)*, *mountfsys(1M)*.
sysadm(1) in the *User's Reference Manual*.



NAME

makehdfsys – construct file systems on hard disks

SYNOPSIS

/usr/admin/menu/diskmgmt/harddisk/makehdfsys

DESCRIPTION

This command, which may be under password control, constructs file systems on hard disks according to device specific information and user response. The command will determine the number of equipped disks and then ask the user which disk to make file systems on. Once the user selects a device, each partition is evaluated for size in blocks, tag, and flag. If the partition size is greater than zero, flag is not set to unmountable or readonly and tag is not set to swap, the partition is checked for an existing file system. If a file system does exist, the user is asked whether the file system is to be erased and another built.

If the partition does not have an existing file system, the user is asked for the label and name of the file system to be constructed. The command then makes a mount point directory if non existent and warns the user of possible problems that may arise if the mount point previously exists. The user is then asked for the maximum number of files and directories for the partition to determine the number of inodes although a default value can be selected. The file system is then constructed and initialized with a lost+found directory.

The user is asked whether the mount point is to be added to the mount table to allow automatic file system checking at power up. At this point the next partition with a block size greater than zero is selected for file system construction.

makehdfsys may be put under password control by the use of **admpasswd** contained on the *sysadm(1)* manual page. The identical function is also available under the *sysadm(1)* command: **sysadm makehdfsys**

The *makehdfsys* command may not be used to make file systems on the host device.

FILES

/usr/sbin/checkyn	interface to user response
/usr/sbin/checklist	interface to user response
/usr/sbin/checkre	interface to user response
/usr/sbin/devinfo	obtains device specific information
/usr/sbin/disklabel	obtains file system label information
/usr/sbin/labelfsname	returns file system label

SEE ALSO

devinfo(1M), *fmthard(1M)*,
prtvtoc(1), *sysadm(1)* in the *User's Reference Manual*.



NAME

`mkboot` – convert an object file to a bootable object file

SYNOPSIS

`/etc/mkboot [-m master] [-d directory] [-k kernel] obj_file`

DESCRIPTION

The `mkboot` command is used to create a bootable object file in a format compatible with the self-configuration program. The object file specified as an argument, `obj_file`, must have a corresponding `master(4)` file in the `/etc/master.d` directory. The master file name for the UNIX system kernel object file is always `kernel`. The other master file names are derived from their associated object file names in lowercase letters minus any optional path prefix or ".o" suffix.

The options are:

- `-m master` This option specifies the directory containing the master files to be used for the object file. The default `master` directory is `/etc/master.d`.
- `-d directory` This option specifies the directory to be used for storing the new bootable object file. The default output `directory` is `/boot`.
- `-k kernel` This option specifies the name of the object file for the UNIX operating system. The master file name used for this object file is always named `kernel`.

To create the new bootable object file, the applicable master file is read and the configuration information is extracted. Then, the new bootable file is created containing this configuration information and written to the output directory specified by the `-d` option or `/boot`. It is given the same name, in uppercase letters and without the ".o" suffix, as the object file. Note that if the current working directory is `/boot` when `mkboot` is executed, then the output object file is the previous bootable object file residing in this directory. This means that you do not have to keep separate ".o" files.

EXAMPLE

`mkboot -m newmaster gentty.o`

This will read the file named `gentty` from the directory `newmaster` for the `gentty` device configuration data, take the file `gentty.o` from the current directory, and create the formatted file `/boot/GENTTY` containing the configuration information for the `gentty`.

`cd /boot; mkboot -k KERNEL`

This will read the file named `kernel` from the directory `/etc/master.d` for the new kernel configuration data, take the file `KERNEL` from the current directory, and create the formatted `/boot/KERNEL` file.

SEE ALSO

`mkunix(1M)`.

`master(4)` in the *Programmer's Reference Manual*.

DIAGNOSTICS

Most messages are self-explanatory.

name.o: not processed; cannot open /etc/master.d/name

The file *name.o* was specified on the command line but there was no master file in the **master.d** directory for *name.o*.

name.o: not processed

An error has aborted processing for the named object file.

NAME

mkfs - construct a file system

SYNOPSIS

```
/etc/mkfs special blocks[:i-nodes] [gap blocks/cyl] [-b blocksize]
/etc/mkfs special proto [gap blocks/cyl] [-b blocksize]
```

DESCRIPTION

mkfs constructs a file system by writing on the *special* file using the values found in the remaining arguments of the command line. The command waits 10 seconds before starting to construct the file system. During this 10-second pause the command can be aborted by entering a delete (DEL).

The *-b blocksize* option specifies the logical block size for the file system. The logical block size is the number of bytes read or written by the operating system in a single I/O operation. Valid values for *blocksize* are 512, 1024, and 2048. The default is 1024. A block size of 2048 may be chosen only if the 2K file system package is installed. If the *-b* option is used it must appear last on the command line.

If the second argument to *mkfs* is a string of digits, the size of the file system is the value of *blocks* interpreted as a decimal number. This is the number of *physical* (512 byte) disk blocks the file system will occupy. If the number of *i-nodes* is not given, the default is approximately the number of *logical* blocks divided by 4. *mkfs* builds a file system with a single empty directory on it. The boot program block (block zero) is left uninitialized.

If the second argument is the name of a file that can be opened, *mkfs* assumes it to be a prototype file *proto*, and will take its directions from that file. The prototype file contains tokens separated by spaces or new-lines. A sample prototype specification follows (line numbers have been added to aid in the explanation):

```
1.    /stand/ diskboot
2.    4872 110
3.    d--777 3 1
4.    usr  d--777 3 1
5.          sh    ---755 3 1 /bin/sh
6.          ken   d--755 6 1
7.          $
8.          b0    b--644 3 1 0 0
9.          c0    c--644 3 1 0 0
10.         $
11.    $
```

Line 1 in the example is the name of a file to be copied onto block zero as the bootstrap program.

Line 2 specifies the number of *physical* (512 byte) blocks the file system is to occupy and the number of *i-nodes* in the file system.

Lines 3-9 tell *mkfs* about files and directories to be included in this file system.

Line 3 specifies the root directory.

Lines 4-6 and 8-9 specifies other directories and files.

The \$ on line 7 tells *mkfs* to end the branch of the file system it is on, and continue from the next higher directory. The \$ on lines 10 and 11 end the process, since no additional specifications follow.

File specifications give the mode, the user ID, the group ID, and the initial contents of the file. Valid syntax for the contents field depends on the first character of the mode.

The mode for a file is specified by a 6-character string. The first character specifies the type of the file. The character range is *-bcd* to specify regular, block special, character special and directory files respectively. The second character of the mode is either *u* or *-* to specify set-user-id mode or not. The third is *g* or *-* for the set-group-id mode. The rest of the mode is a 3 digit octal number giving the owner, group, and other read, write, execute permissions (see *chmod(1)*).

Two decimal number tokens come after the mode; they specify the user and group IDs of the owner of the file.

If the file is a regular file, the next token of the specification may be a path name whence the contents and size are copied. If the file is a block or character special file, two decimal numbers follow which give the major and minor device numbers. If the file is a directory, *mkfs* makes the entries *.* and *..* and then reads a list of names and (recursively) file specifications for the entries in the directory. As noted above, the scan is terminated with the token \$.

The *gap blocks/cyl* argument in both forms of the command specifies the rotational gap and the number of blocks/cylinder. The following values are recommended for the devices available on the 3B2:

Device	Gap Size 512-byte FS	Gap Size 1K FS	Gap Size 2K FS	Blks/Cyl	
10M Hard Disk	8	10	12	72	
30M Hard Disk	8	10	12	90	
72M Hard Disk	8	10	12	162	(CDC Wren II)
72aM Hard Disk	8	10	12	144	(Micropolis)
72bM Hard Disk	8	10	12	162	(Priam)
72cM Hard Disk	8	10	12	198	(Fujitsu)
Floppy Disk	1	1	1	18	

If the *gap* and *blocks/cyl* are not specified or are considered illegal values a default value of gap size 7 and 400 blocks/cyl is used.

FILES

/etc/vtoc/*

SEE ALSO

dir(4), *fs(4)*.
chmod(1) in the *User's Reference Manual*.

BUGS

With a prototype file, it is not possible to copy in a file larger than 64K bytes, nor is there a way to specify links. The maximum number of i-nodes configurable is 65500.

NAME

`mknod` – build special file

SYNOPSIS

```
/etc/mknod name b | c major minor  
/etc/mknod name p
```

DESCRIPTION

`mknod` makes a directory entry and corresponding i-node for a special file.

The first argument is the *name* of the entry. The UNIX System convention is to keep such files in the `/dev` directory.

In the first case, the second argument is `b` if the special file is block-type (disks, tape) or `c` if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g., unit, drive, or line number). They may be either decimal or octal. The assignment of major device numbers is specific to each system. The information is contained in the system source file `conf.c`. You must be the super-user to use this form of the command.

The second case is the form of the `mknod` that is used to create FIFO's (a.k.a named pipes).

WARNING

If `mknod` is used to create a device in a remote directory (Remote File Sharing), the major and minor device numbers are interpreted by the server.

SEE ALSO

`mknod(2)` in the *Programmer's Reference Manual*.



NAME

`mkunix` – make a bootable system file with kernel and driver symbol tables

SYNOPSIS

```
/etc/mkunix [ system_namelist ] [-o new_namelist ]
```

DESCRIPTION

The `mkunix` command will create an absolute, bootable system file (`new_namelist`) from the UNIX system kernel file and the object files that were loaded by the self-configuration program. In effect, this procedure completes the generation of a new `/unix`. Typically, `mkunix` would be run following an auto-configuration boot with a new system configuration. It can only be used by the super-user.

The resulting `new_namelist` can be used as the `system_namelist` for `ps`, `crash`, etc. In addition, this file may be booted directly, bypassing the self-configuration phase of the boot process (see `3b2boot`[8]). This will save on the order of 30 to 60 seconds at boot time.

`System_namelist` (defaults to the path name specified as the `BOOT` program in the `/etc/system` file) is read to obtain the object, data, and symbol table for the basic kernel. This name, if specified, must be the same as that used in `/etc/system` for the boot line; if not, a warning diagnostic is issued since the resulting namelist file will not be accurate.

The argument `-o new_namelist` (defaults to `a.out`) is the new file--a bootable image of the currently running UNIX system with the composite symbol table.

SEE ALSO

`crash`(1M), `mkboot`(1M), `3b2boot`(8).
`nm`(1), `ps`(1) in the *User's Reference Manual*.



NAME

mount, umount – mount and unmount file systems and remote resources

SYNOPSIS

```
/etc/mount [-r] [-f fstyp] special directory
/etc/mount [-r] [-c] -d resource directory
/etc/mount
/etc/umount special
/etc/umount -d resource
```

DESCRIPTION

File systems other than **root** (/) are considered *removable* in the sense that they can be either available to users or unavailable. *mount* announces to the system that *special*, a block special device or *resource*, a remote resource, is available to users from the mount point *directory*. *directory* must exist already; it becomes the name of the root of the newly mounted *special* or *resource*. A unique resource may be mounted only once (no multiple mounts).

mount, when entered with arguments, adds an entry to the table of mounted devices, */etc/mnttab*. *umount* removes the entry. If invoked with no arguments, *mount* prints the entire mount table. If invoked with any of the following partial argument lists, *mount* will search */etc/fstab* to fill in the missing arguments: *special*, **-d resource**, *directory*, or **-d directory**.

The following options are available:

- r** indicates that *special* or *resource* is to be mounted read-only. If *special* or *resource* is write-protected or read-only advertised, this flag must be used.
- d** indicates that *resource* is a remote resource that is to be mounted on *directory* or unmounted. To mount a remote resource, Remote File Sharing must be up and running and the resource must be advertised by a remote computer [see *rfstart*(1M) and *adv*(1M)]. If **-d** is not used, *special* must be a local block special device.
- c** indicates that remote reads and writes should not be cached in the local buffer pool. **-c** is used in conjunction with **-d**.
- f fstyp** indicates that *fstyp* is the file system type to be mounted. If this argument is omitted, it defaults to the **root** *fstyp*.
- special* indicates the block special device that is to be mounted on *directory*.
- resource* indicates the remote resource name that is to be mounted on a *directory*.
- directory* indicates the directory mount point for *special* or *resource*. (The *directory* must already exist.)

umount announces to the system that the previously mounted *special* or *resource* is to be made unavailable. If invoked with *directory* or **-d directory**, *umount* will search */etc/fstab* to fill in the missing argument(s).

mount can be used by any user to list mounted file systems and resources. Only a super-user can mount and unmount file systems.

FILES

/etc/mnttab	mount table
/etc/fstab	file system table

SEE ALSO

adv(1M), fuser(1M), nsquery(1M), rfstart(1M), rmntstat(1M), setmnt(1M), unadv(1M), fstab(4), mnttab(4).
mount(2), umount(2) in the *Programmer's Reference Manual*.
"Remote File Sharing" chapter, *System Administrator's Guide*, for guidelines on mounting remote resources.

DIAGNOSTICS

If the *mount(2)* system call fails, *mount* prints an appropriate diagnostic. *mount* issues a warning if the file system to be mounted is currently labeled under another name. A remote resource mount will fail if the resource is not available or if Remote File Sharing is not running or if it is advertised read-only and not mounted with *-r*.

umount fails if *special* or *resource* is not mounted or if it is busy. *special* or *resource* is busy if it contains an open file or some user's working directory. In such a case, you can use *fuser(1M)* to list and kill processes that are using *special* or *resource*.

WARNINGS

Physically removing a mounted file system diskette from the diskette drive before issuing the *umount* command damages the file system.

NAME

mountall, umountall – mount, unmount multiple file systems

SYNOPSIS

```
/etc/mountall [-] [file-system-table] ...
/etc/umountall [-k]
```

DESCRIPTION

These commands may be executed only by the super-user.

mountall is used to mount file systems according to a *file-system-table*. (*/etc/fstab* is the default file system table.) The special file name "-" reads from the standard input.

Before each file system is mounted, it is checked using *fsstat*(1M) to see if it appears mountable. If the file system does not appear mountable, it is checked, using *fsck*(1M), before the mount is attempted.

umountall causes all mounted file systems except **root** to be unmounted. The **-k** option sends a SIGKILL signal, via *fuser*(1M), to processes that have files open.

FILES

File-system-table format:

column 1	block special file name of file system
column 2	mount-point directory
column 3	-r if to be mounted read-only; -d if remote
column 4	(optional) file system type string
column 5+	ignored

White-space separates columns. Lines beginning with "#" are comments. Empty lines are ignored.

A typical *file-system-table* might read:

```
/dev/dsk/c1d0s2 /usr -r S51K
```

SEE ALSO

fsck(1M), *fsstat*(1M), *fuser*(1M), *mount*(1M), *fstab*(4).
sysadm(1) in the *User's Reference Manual*.
signal(2) in the *Programmer's Reference Manual*.

DIAGNOSTICS

No messages are printed if the file systems are mountable and clean.

Error and warning messages come from *fsck*(1M), *fsstat*(1M), and *mount*(1M).



NAME

mountfsys, umountfsys – mount, unmount a diskette file system

SYNOPSIS

```
mountfsys [ -y ] [ -r ]
umountfsys [ -y ]
```

DESCRIPTION

The *mountfsys* command mounts a file system that is on a removable disk so that users can read and write on it. The options provide the following:

- r the file system is mounted read-only.
- y suppresses any questions asked during mounting or unmounting.

The *umountfsys* command unmounts the file system.

By default, the name of the file system is displayed and the user is asked if it should be mounted. The optional *–y* argument suppresses questions and mounts or unmounts the file system immediately.

The identical functions are available under the *sysadm* menu:

```
sysadm mountfsys
sysadm umountfsys
```

The commands may be assigned passwords. See *sysadm(1)*, the **admpasswd** sub-command.

SEE ALSO

checkfsys(1M), *mount(1M)*, *makefsys(1M)*.
sysadm(1) in the *User's Reference Manual*.

WARNING

ONCE THE DISK IS MOUNTED IT MUST NOT BE REMOVED FROM THE DISK DRIVE UNTIL IT HAS BEEN UNMOUNTED!

Removing the disk while it is still mounted can cause severe damage to the data on the disk.

BUGS

A file system that has no label cannot be mounted with the *mountfsys* command.



NAME

`mmdir` – move a directory

SYNOPSIS

`/etc/mmdir` *dirname* *name*

DESCRIPTION

mmdir moves directories within a file system. *Dirname* must be a directory. If *name* does not exist, it will be created as a directory. If *name* does exist, and is a directory, *dirname* will be created as *name/dirname*. *Dirname* and *name* may not be on the same path; that is, one may not be subordinate to the other. For example:

`mmdir x/y x/z`

is legal, but

`mmdir x/y x/y/z`

is not.

SEE ALSO

`mkdir(1)`, `mv(1)` in the *User's Reference Manual*.

WARNINGS

Only the super-user can use *mmdir*.



NAME

ncheck – generate path names from i-numbers

SYNOPSIS

`/etc/ncheck [-i i-numbers] [-a] [-s] [file-system]`

DESCRIPTION

ncheck with no arguments generates a path-name vs. i-number list of all files on a set of default file systems (see */etc/checklist*). Names of directory files are followed by *./*.

The options are as follows:

- i** limits the report to only those files whose i-numbers follow.
- a** allows printing of the names *.* and *..*, which are ordinarily suppressed.
- s** limits the report to special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

File system must be specified by the file system's special file.

The report should be sorted so that it is more useful.

SEE ALSO

fsck(1M).
sort(1) in the *User's Reference Manual*.

DIAGNOSTICS

If the file system structure is not consistent, *??* denotes the "parent" of a parentless file and a path-name beginning with *...* denotes a loop.



NAME

`newboot` – load `olboot` and `mboot` onto the disk boot partition

SYNOPSIS

`/etc/newboot [-y] olboot mboot boot_special`

DESCRIPTION

`newboot` places the `olboot` and `mboot` files on the given `boot_special` section of the disk (`c1d0s7`). If the `-y` option is not specified, you are prompted for a confirmation before the `boot_special` file is overwritten.

`Mboot` is the 512-byte micro-boot file loaded by the boot device firmware and executed to load the larger `lboot` file.

`Olboot` is a file containing the boot program that is loaded by `mboot` and executed to boot the absolute operating system.

SEE ALSO

`dd(1M)`, `mkboot(1M)`.

DIAGNOSTICS

Can't open file

The boot file `file` was not found.

WARNINGS

Installing a bad `olboot` or `mboot` may make the affected disk unbootable. Be sure you have a good backup copy of the disk before `newboot` is run.



NAME

`newgrp` – log in to a new group

SYNOPSIS

`newgrp [-] [group]`

DESCRIPTION

`newgrp` changes a user's group identification. The user remains logged in and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new real and effective group IDs. The user is always given a new shell, replacing the current shell, by `newgrp`, regardless of whether it terminated successfully or due to an error condition (i.e., unknown group).

Exported variables retain their values after invoking `newgrp`; however, all unexported variables are either reset to their default value or set to null. System variables (such as `PS1`, `PS2`, `PATH`, `MAIL`, and `HOME`), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (`PS1`) other than `$` (default) and has not exported `PS1`. After an invocation of `newgrp`, successful or not, their `PS1` will now be set to the default prompt string `$`. Note that the shell command `export` (see `sh(1)`) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, `newgrp` changes the group identification back to the group specified in the user's password file entry. This is a way to exit the effect of an earlier `newgrp` command.

If the first argument to `newgrp` is a `-`, the environment is changed to what would be expected if the user actually logged in again as a member of the new group.

A password is demanded if the group has a password and the user does not, or if the group has a password and the user is not listed in `/etc/group` as being a member of that group.

FILES

<code>/etc/group</code>	system's group file
<code>/etc/passwd</code>	system's password file

SEE ALSO

`login(1)`, `sh(1)` in the *User's Reference Manual*.

`group(4)`, `passwd(4)`, `environ(5)` in the *System Administrator's Reference Manual*.

BUGS

There is no convenient way to enter a password into `/etc/group`. Use of group passwords is not encouraged, because, by their very nature, they encourage poor security practices. Group passwords may disappear in the future.



NAME

`nlsadmin` – network listener service administration

SYNOPSIS

```
nlsadmin -x
nlsadmin [ options ] net_spec
```

DESCRIPTION

nlsadmin administers the network listener process(es) on a machine. Each network has a separate instance of the network listener process associated with it; each instance (and thus, each network) is configured separately. The listener process "listens" to the network for service requests, accepts requests when they arrive, and spawns servers in response to those service requests. The network listener process will work with any network (more precisely, with any transport provider) that conforms to the transport provider specification.

The listener supports two classes of service: a general listener service, serving processes on remote machines, and a terminal login service, for terminals connected directly to a network. The terminal login service provides networked access to this machine in a form suitable for terminals connected directly to the network. However, this direct terminal service requires special associated software, and is only available with some networks (for example, the AT&T STARLAN network).

nlsadmin can establish a listener process for a given network, configure the specific attributes of that listener, and start and kill the listener process for that network. *nlsadmin* can also report on the listener processes on a machine, either individually (per network) or collectively.

The following list shows how to use *nlsadmin*. In this list, *net_spec* represents a particular listener process. Specifically, *net_spec* is the relative path name of the entry under `/dev` for a given network (that is, a transport provider). Changing the list of services provided by the listener produces immediate changes, while changing an address on which the listener listens has no effect until the listener is restarted. The following combination of options can be used.

- | | |
|-----------------------------|--|
| nlsadmin | will give a brief usage message. |
| nlsadmin -x | will report the status of all of the listener processes installed on this machine. |
| nlsadmin net_spec | will print the status of the listener process for <i>net_spec</i> . |
| nlsadmin -q net_spec | will query the status of the listener process for the specified network, and will reflect the result of that query in its exit code. If a listener process is active, <i>nlsadmin</i> will exit with a status of 0; if no process is active, the exit code will be 1; the exit code will be greater than 1 in case of error. |
| nlsadmin -v net_spec | will print a verbose report on the servers associated with <i>net_spec</i> , giving the service code, status, command, and comment for each. It also specifies the uid the |

server will run as, and the list of modules to be pushed, if any, before the server is started.

nlsadmin -z *service_code net_spec*

will print a report on the server associated with *net_spec* that has service code *service_code*, giving the same information as in the **-v** option.

nlsadmin -q -z *service_code net_spec*

will query the status of the service with service code *service_code* on network *net_spec*, and will exit with a status of 0 if that service is enabled, 1 if that service is disabled, and greater than 1 in case of error.

nlsadmin -l *addr net_spec*

will change or set the address on which the listener listens (the general listener service). This is the address generally used by remote processes to access the servers available through this listener (see the **-a** option, below). *addr* is the transport address on which to listen and is interpreted using a syntax that allows for a variety of address formats. By default *addr* is interpreted as the symbolic ASCII representation of the transport address. An *addr* preceded by a **\x** will let you enter an address in hexadecimal notation. Note that *addr* must appear as a single word to the shell and must be quoted if it contains any blanks.

If *addr* is just a dash ("-"), *nlsadmin* will report the address currently configured, instead of changing it.

A change of address will not take effect until the next time the listener for that network is started.

nlsadmin -t *addr net_spec*

will change or set the address on which the listener listens for requests for terminal service, but is otherwise similar to the **-l** option above. A terminal service address should not be defined unless the appropriate remote login software is available; if such software is available, it must be configured as service code 1 (see the **-a** option, below).

nlsadmin -i *net_spec*

will initialize or change a listener process for the network specified by *net_spec*, that is, it will create and initialize the files required by the listener. Note that the listener should only be initialized once for a given network, and that doing so does not actually invoke the listener for that network. The listener must be initialized before assigning addressing or services.

nlsadmin [-m] -a *service_code [-p modules] [-w id] -c cmd -y comment net_spec*

will add a new service to the list of services available through the indicated listener. *service_code* is the code

for the service, *cmd* is the command to be invoked in response to that service code, comprised of the full path name of the server and its arguments, and *comment* is a brief (free-form) description of the service for use in various reports. Note that *cmd* must appear as a single word to the shell, so if arguments are required the *cmd* and its arguments must be surrounded by quotes. Similarly, the *comment* must also appear as a single word to the shell. When a service is added, it is initially enabled (see the `-e` and `-d` options, below).

If the `-m` option is specified, the entry will be marked as an administrative entry. Service codes 1 through 100 are reserved for administrative entries, which are those that require special handling internally. In particular, code 1 is assigned to the remote login service, which is the service automatically invoked for connections to the terminal login address.

The `-m` option used with the `-a` option indicates that special handling internally is required for those servers added with the `-m` set. This internal handling is in the form of code embedded on the listener process.

If the `-p` option is specified, then *modules* will be interpreted as a list of STREAMS modules for the listener to push before starting the service being added. The modules are pushed in the order they are specified. *modules* should be a comma-separated list of modules, with no white space included.

If the `-w` option is specified, then *id* is interpreted as the user name from `/etc/passwd` that the listener should look up. From the user name, the listener should obtain the user ID, the group ID, and the home directory for use by the server. If `-w` is not specified, the default is to use the user ID `listen`.

A service must explicitly be added to the listener for each network on which that service is to be available. This operation will normally be performed only when the service is installed on a machine, or when populating the list of services for a new network.

nlsadmin -r *service_code net_spec*

will remove the entry for the *service_code* from that listener's list of services. This will normally be performed only in conjunction with the de-installation of a service from a machine.

nlsadmin -e *service_code net_spec*

nlsadmin -d *service_code net_spec*

will enable or disable (respectively) the service indicated by *service_code* for the specified network. The service must have previously been added to the listener for that network (see the **-a** option, above). Disabling a service will cause subsequent service requests for that service to be denied, but the processes from any prior service requests that are still running will continue unaffected.

nlsadmin -s *net_spec*

nlsadmin -k *net_spec*

will start and kill (respectively) the listener process for the indicated network. These operations will normally be performed as part of the system startup and shutdown procedures. Before a listener can be started for a particular network, it must first have been initialized, and an address must be defined for the general listener service (see the **-i** and **-l** options, above). When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected.

The listener runs as user ID **root**, with group ID **sys**. A special ID, user ID **listen** and group ID **adm**, should be entered in the **/etc/passwd** file as a default ID for servers. The listener always uses as its home directory **/usr/net/nls**, which is concatenated with *net_spec* to determine the location of the listener configuration information for each network. The home directory specified in the **/etc/passwd** entry for **listen** will be used by servers that run as ID **listen**.

nlsadmin may be invoked by any user to generate reports, but all operations that affect a listener's status or configuration are restricted to the super-user.

FILES

/usr/net/nls/net_spec

SEE ALSO

Network Programmer's Guide

NAME

nsquery – Remote File Sharing name server query

SYNOPSIS

nsquery [-h] [*name*]

DESCRIPTION

nsquery provides information about resources available to the host from both the local domain and from other domains. All resources are reported, regardless of whether the host is authorized to access them. When used with no options, *nsquery* identifies all resources in the domain that have been advertised as sharable. A report on selected resources can be obtained by specifying *name*, where *name* is:

nodename The report will include only those resources available from *nodename*.

domain. The report will include only those resources available from *domain*.

domain.nodename The report will include only those resources available from *domain.nodename*.

When the name does not include the delimiter ".", it will be interpreted as a *nodename* within the local domain. If the name ends with a delimiter ".", it will be interpreted as a domain name.

The information contained in the report on each resource includes its advertised name (domain.resource), the read/write permissions, the server (nodename.domain) that advertised the resource, and a brief textual description.

When *-h* is used, the header is not printed.

A remote domain must be listed in your **rfmaster** file in order to query that domain.

EXIT STATUS

If no entries are found when *nsquery* is executed, the report header is printed.

ERRORS

If your host cannot contact the domain name server, an error message will be sent to standard error.

SEE ALSO

adv(1M), **unadv(1M)**, **rfmaster(4)**.



NAME

passmgmt – password files management

SYNOPSIS

passmgmt –*a options name*
passmgmt –*m options name*
passmgmt –*d name*

DESCRIPTION

The **passmgmt** command updates information in the password files. This command works with both **/etc/passwd** and **/etc/shadow**. If there is no **/etc/shadow**, the changes done by **passmgmt** will only go to **/etc/passwd**.

passmgmt –*a* adds an entry for user *name* to the password files. This command does not create any directory for the new user and the new login remains locked (with the string **LK** in the password field) until the **passwd(1)** command is executed to set the password.

passmgmt –*m* modifies the entry for user *name* in the password files. The name field in the **/etc/shadow** entry and all the fields (except the password field) in the **/etc/passwd** entry can be modified by this command. Only fields entered on the command line will be modified.

passmgmt –*d* deletes the entry for user *name* from the password files. It will not remove any files that the user owns on the system; they must be removed manually.

The following options are available:

- c comment* A short description of the login. It is limited to a maximum of 128 characters and defaults to an empty field.
- h homedir* Home directory of *name*. It is limited to a maximum of 256 characters and defaults to **/usr/name**.
- u uid* UID of the *name*. This number must range from 0 to the maximum non-negative value for the system. It defaults to the next available UID greater than 99. Without the –*o* option, it enforces the uniqueness of a UID.
- o* This option allows a UID to be non-unique. It is used only with the –*u* option.
- g gid* GID of the *name*. This number must range from 0 to the maximum non-negative value for the system. The default is 1.
- s shell* Login shell for *name*. It should be the full pathname of the program that will be executed when the user logs in. The maximum size of *shell* is 256 characters. The default is for this field to be empty and to be interpreted as **/bin/sh**.
- l logname* This option changes the *name* to *logname*. It is used only with the –*m* option.

The total size of each login entry is limited to a maximum of 511 bytes in each of the password files.

FILES

/etc/passwd, /etc/shadow, /etc/opasswd, /etc/oshadow

SEE ALSO

passwd(4) in the *System Administrator's Reference Manual*.
passwd(1) in the *User's Reference Manual*.

DIAGNOSTICS

The **passmgmt** command exits with one of the following values:

- 0 SUCCESS.
- 1 Permission denied.
- 2 Invalid command syntax. Usage message of the **passmgmt** command will be displayed.
- 3 Invalid argument provided to option.
- 4 UID in use.
- 5 Inconsistent password files (e.g., *name* is in the **/etc/passwd** file and not in the **/etc/shadow** file, or vice versa).
- 6 Unexpected failure. Password files unchanged.
- 7 Unexpected failure. Password file(s) missing.
- 8 Password file(s) busy. Try again later.
- 9 *name* does not exist (if **-m** or **-d** is specified), already exists (if **-a** is specified), or *logname* already exists (if **-m -l** is specified).

NOTE

You cannot use a colon or <cr> as part of an argument because it will be interpreted as a field separator in the password file.

NAME

powerdown – stop all processes and turn off the power

SYNOPSIS

powerdown [**-y** | **-Y**]

DESCRIPTION

This command brings the system to a state where nothing is running and then turns off the power.

By default, the user is asked questions that control how much warning the other users are given. The options:

- y** prevents the questions from being asked and just gives the warning messages. There is a 60 second pause between the warning messages. Note that pressing the standby button on the side of the cabinet will accomplish the same thing.
- Y** is the same as **-y** except it has no pause between messages. It is the fastest way to bring the system down.

The identical function is also available under the *sysadm* command:

sysadm powerdown

Password control can be instituted on this command. See *sysadm(1)*, **admpasswd** sub-command.

EXAMPLES

some-long-running-command; powerdown **-y**

The first command is run to completion and then the machine turns off. This is useful for, say, formatting a document to the printer at the end of a day.

FILES

/etc/shutdown - invoked by powerdown

SEE ALSO

shutdown(1M).

sysadm(1) in the *User's Reference Manual*.



NAME

profiler: *prfld*, *prfstat*, *prfdc*, *prfsnap*, *prfpr* – UNIX system profiler

SYNOPSIS

```

/etc/prfld [ system_namelist ]
/etc/prfstat on
/etc/prfstat off
/etc/prfdc file [ period [ off_hour ] ]
/etc/prfsnap file
/etc/prfpr file [ cutoff [ system_namelist ] ]

```

DESCRIPTION

Prfld, *prfstat*, *prfdc*, *prfsnap*, and *prfpr* form a system of programs to facilitate an activity study of the UNIX operating system.

Prfld is used to initialize the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as extracted from *system_namelist*.

Prfstat is used to enable or disable the sampling mechanism. Profiler overhead is less than 1% as calculated for 500 text addresses. *Prfstat* will also reveal the number of text addresses being measured.

Prfdc and *prfsnap* perform the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. *Prfdc* will store the counters into *file* every *period* minutes and will turn off at *off_hour* (valid values for *off_hour* are 0–24). *Prfsnap* collects data at the time of invocation only, appending the counter values to *file*.

Prfpr formats the data collected by *prfdc* or *prfsnap*. Each text address is converted to the nearest text symbol (as found in *system_namelist*) and is printed if the percent activity for that range is greater than *cutoff*.

FILES

```

/dev/prf      interface to profile data and text addresses
/unix        default for system namelist file

```



NAME

`prtconf` – print system configuration

SYNOPSIS

`/etc/prtconf`

DESCRIPTION

The `prtconf` command prints the system configuration information which includes the memory and peripheral configuration. This information is displayed every time the system is initialized to multiuser mode.

EXAMPLES

To print the configuration of the 3B2 Computer, execute:

`/etc/prtconf<CR>`

AT&T 3B2 SYSTEM CONFIGURATION:

Memory size: 2 Megabytes

System Peripherals:

Device Name	Subdevices	Extended Subdevices
SBD	Floppy Disk 30 Megabyte Disk 72 Megabyte Disk	
SCSI	SD00 ID1	67 Megabyte Disk ID0 67 Megabyte Disk ID1 135 Megabyte Disk ID2 135 Megabyte Disk ID3
NI		
PORTS		
CTC		



NAME

`prvtoc` – print the VTOC of a block device

SYNOPSIS

`/etc/prvtoc device`

DESCRIPTION

The `prvtoc` command allows the contents of the VTOC (volume table of contents) to be viewed. The command can be used only by the super-user.

The `device` name must be the file name of a raw device in the form of `/dev/rdisk/c?t?s?` or `/dev/rdisk/c?t?d?s6?`.

EXAMPLE

The command line entry and system response shown below are for a 135-megabyte hard disk:

```
# prvtoc /dev/rdisk/c2t1d0s6 <CR>
* /dev/rdisk/c2t1d0s6 partition map
*
* Dimension:
*   512 bytes/sector
*   34 sectors/track
*   10 tracks/cylinder
*   340 sectors/cylinder
*   816 cylinders
*   814 accessible cylinders
*
* Flags:
*   1: unmountable
*   10: read-only
*
*
* Partition   Tag   Flags   First Sector   Sector Count   Last Sector   Mount Directory
*   0         2     00     19040         23460         42499         /
*   1         3     01         100         18940         19039
*   2         4     00     42500         234260        276759        /usr
*   6         0     01          0         276760        276759
*   7         0     01          0          100          99
*
#
```

Codes for TAG are:

NAME	NUMBER
UNASSIGNED	0
BOOT	1
ROOT	2
SWAP	3
USR	4
BACKUP	5

FLAG indicates how the partition is to be mounted.

NAME	NUMBER
MOUNTABLE, READ AND WRITE	00
NOT MOUNTABLE	01
MOUNTABLE, READ ONLY	10

SEE ALSO

fmthard(1M).

CAVEAT

The **mount** command does not check the "not mountable" bit.

NAME

pump – Download B16 or X86 a.out file to a peripheral board

SYNOPSIS

/etc/pump /dev/devname file

DESCRIPTION

The *pump* command will read a B16 or X86 **a.out** file's sections into a buffer according to the physical address of the section. The command can be used only by the super-user. *pump* expects a section in the **a.out** file called ".start". Once it has found this section, *pump* will inform the peripheral to start executing at the address that it found in ".start" after it has downloaded the **a.out** file.

There are four phases of the pump operation:

- reset** This phase will cause a hardware reset on the peripheral.
- download** This phase will download the **a.out** file to the peripheral.
- force call to function**
 This phase will inform the peripheral to start executing at the address found in the ".start" section.
- sysgen** This phase will *sysgen* the peripheral. It allows normal functioning of the peripheral to occur.

Error Messages

pump error: UNIX_error_number – Can't get status of /dev/devname

There may be no **/dev/devname**.

pump Error: error_number – ioctl call

The ioctl call failed. The *error_number* returned can be a UNIX system error number or, in the case of the NI, an error number of 208. Error number 208 is a timeout message. The peripheral board did not respond in time to the request made of it (this is not the only error, see *intro(2)* for a complete list).

Can't open a.out filename for reading!

The error may be that there is no such file or the permissions are such that the file cannot be read [see *chmod(1)*].

Error: Object file is not in b16 or x86 common object format

The file to be downloaded to the peripheral is not a B16 or X86 **a.out** file.

Section size is too big for the buffer

The **a.out** file may be greater than the 32K bytes that is the limit of RAM on the peripheral.

Error: No section name called .start

pump needs ".start" for the starting address that the peripheral needs to execute the downloaded code.

pump: /dev/devname returned a CIO FAULT during phase

The peripheral encountered a hardware fault during one of the phases of the pump.

pump: /dev/devname returned a CIO Invalid Queue Entry during phase

The peripheral did not understand the command phase that was issued by *pump*.

pump: /dev/devname did not respond during phase

The UNIX system driver called may not have understood the command.

pump: A timeout has occurred on /dev/devname during phase

The peripheral did not respond to one of the commands given.

pump: There was no return code for /dev/devname during phase

The return code that was given may have been corrupted.

SEE ALSO

intro(2), a.out(4).

NAME

pwck, *grpck* – password/group file checkers

SYNOPSIS

/etc/pwck [file]
/etc/grpck [file]

DESCRIPTION

pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-Shell exist. The default password file is */etc/passwd*.

Grpck verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is */etc/group*.

FILES

/etc/group
/etc/passwd

SEE ALSO

group(4), *passwd(4)*.

DIAGNOSTICS

Group entries in */etc/group* with no login names are flagged.



NAME

`pwconv` - Installs and updates `/etc/shadow` with information from `/etc/passwd`

SYNOPSIS

`pwconv`

DESCRIPTION

The `pwconv` command creates and updates `/etc/shadow` with information from `/etc/passwd`.

If the `/etc/shadow` file does not exist, this command will create `/etc/shadow` with information from `/etc/passwd`. The command populates `/etc/shadow` with the user's login name, password, and password aging information. If password aging information does not exist in `/etc/passwd` for a given user, none will be added to `/etc/shadow`. However, the "last changed" information will always be updated.

If the `/etc/shadow` file does exist, the following tasks will be performed:

Entries that are in the `/etc/passwd` file and not in the `/etc/shadow` file will be added to the `/etc/shadow` file.

Entries that are in the `/etc/shadow` file and not in the `/etc/passwd` file will be removed from `/etc/shadow`.

Password attributes (e.g., password and aging information) that exist in an `/etc/passwd` entry will be moved to the corresponding entry in `/etc/shadow`.

The `pwconv` program is a privileged system command that cannot be executed by ordinary users.

FILES

`/etc/passwd`, `/etc/shadow`, `/etc/opasswd`, `/etc/oshadow`

SEE ALSO

`passwd(1)`, `passmgmt(1M)`

DIAGNOSTICS

The `pwconv` command exits with one of the following values:

- | | |
|---|---|
| 0 | SUCCESS. |
| 1 | Permission denied. |
| 2 | Invalid command syntax. |
| 3 | Unexpected failure. Conversion not done. |
| 4 | Unexpected failure. Password file(s) missing. |
| 5 | Password file(s) busy. Try again later. |



NAME

`pwunconv` – Converts from a two- to a one-password file scheme.

SYNOPSIS

`pwunconv`

DESCRIPTION

Pwunconv converts a UNIX system from a two password file scheme (`/etc/passwd` and `/etc/shadow`) to a one password file scheme (`/etc/passwd`). It updates `/etc/passwd` with password information from `/etc/shadow`. If aging information is present in `/etc/shadow`, the password aging information in `/etc/passwd` is also updated.

The total size of a login entry for the password file is limited to a maximum of 511 bytes.

FILES

`/etc/passwd`, `/etc/shadow`, `/etc/opasswd`, `/etc/oshadow`

SEE ALSO

`passwd(1M)`, `passmgmt(1M)`, `pwconv(1M)`

DIAGNOSTICS

The `pwunconv` command exits with one of the following values:

- | | |
|---|--|
| 0 | SUCCESS. |
| 1 | Permission denied. |
| 2 | Invalid command syntax. |
| 3 | Unexpected failure. Conversion not done. |
| 4 | Unexpected failure. Password file missing. |
| 5 | Password file(s) busy. Try again later. |
| 6 | Shadow password file does not exist. |
| 7 | Entry for <i>name</i> too long. Conversion not done. |



NAME

rc0 – run commands performed to stop the operating system

SYNOPSIS

`/etc/rc0`

DESCRIPTION

This file is executed at each system state change that needs to have the system in an inactive state. It is responsible for those actions that bring the system to a quiescent state, traditionally called “shutdown”.

There are three system states that require this procedure. They are state 0 (the system halt state), state 5 (the firmware state), and state 6 (the reboot state). Whenever a change to one of these states occurs, the `/etc/rc0` procedure is run. The entry in `/etc/inittab` might read:

```
s0:056:wait:/etc/rc0 > /dev/console 2>&1 </dev/console
```

Some of the actions performed by `/etc/rc0` are carried out by files in the directory `/etc/shutdown.d.` and files beginning with **K** in `/etc/rc0.d.` These files are executed in ascii order (see FILES below for more information), terminating some system service. The combination of commands in `/etc/rc0` and files in `/etc/shutdown.d` and `/etc/rc0.d` determines how the system is shut down.

The recommended sequence for `/etc/rc0` is:

Stop System Services and Daemons.

Various system services (such as 3BNET Local Area Network or LP Spooler) are gracefully terminated.

When new services are added that should be terminated when the system is shut down, the appropriate files are installed in `/etc/shutdown.d` and `/etc/rc0.d.`

Terminate Processes

SIGTERM signals are sent to all running processes by `killall(1M).` Processes stop themselves cleanly if sent SIGTERM.

Kill Processes

SIGKILL signals are sent to all remaining processes; no process can resist SIGKILL.

At this point the only processes left are those associated with `/etc/rc0` and processes 0 and 1, which are special to the operating system.

Unmount All File Systems

Only the root file system (`/`) remains mounted.

Depending on which system state the systems end up in (0, 5, or 6), the entries in `/etc/inittab` will direct what happens next. If the `/etc/inittab` has not defined any other actions to be performed as in the case of system state 0,

then the operating system will have nothing to do. It should not be possible to get the system's attention. The only thing that can be done is to turn off the power or possibly get the attention of a firmware monitor. The command can be used only by the super-user.

FILES

The execution by `/bin/sh` of any files in `/etc/shutdown.d` occurs in ascii sort-sequence order. See `rc2(1M)` for more information.

SEE ALSO

`killall(1M)`, `rc2(1M)`, `shutdown(1M)`.

NAME

rc2 – run commands performed for multi-user environment

SYNOPSIS

/etc/rc2

DESCRIPTION

This file is executed via an entry in */etc/inittab* and is responsible for those initializations that bring the system to a ready-to-use state, traditionally state 2, called the "multi-user" state.

The actions performed by */etc/rc2* are found in files in the directory */etc/rc.d* and files beginning with *S* in */etc/rc2.d*. These files are executed by */bin/sh* in ascii sort-sequence order (see FILES for more information). When functions are added that need to be initialized when the system goes multi-user, an appropriate file should be added in */etc/rc2.d*.

The functions done by */etc/rc2* command and associated */etc/rc2.d* files include:

Setting and exporting the TIMEZONE variable.

Setting-up and mounting the user (*/usr*) file system.

Cleaning up (remaking) the */tmp* and */usr/tmp* directories.

Loading the network interface and ports cards with program data and starting the associated processes.

Starting the *cron* daemon by executing */etc/cron*.

Cleaning up (deleting) uucp locks status, and temporary files in the */usr/spool/uucp* directory.

Other functions can be added, as required, to support the addition of hardware and software features.

EXAMPLES

The following are prototypical files found in */etc/rc2.d*. These files are prefixed by an *S* and a number indicating the execution order of the files.

MOUNTFILESYS

```
# Set up and mount file systems
```

```
cd /  
/etc/mountall /etc/fstab
```

RMTMPFILES

```
# clean up /tmp  
rm -rf /tmp  
mkdir /tmp  
chmod 777 /tmp  
chgrp sys /tmp  
chown sys /tmp
```

```
uucp
# clean-up uucp locks, status, and temporary files

rm -rf /usr/spool/locks/*
```

The file `/etc/TIMEZONE` is included early in `/etc/rc2`, thus establishing the default time zone for all commands that follow.

FILES

Here are some hints about files in `/etc/rc.d`:

The order in which files are executed is important. Since they are executed in `ascii sort—sequence` order, using the first character of the file name as a sequence indicator will help keep the proper order. Thus, files starting with the following characters would be:

```
[0-9].  very early
[A-Z].  early
[a-n].  later
[o-z].  last
```

3.mountfs

Files in `/etc/rc.d` that begin with a dot (`.`) will not be executed. This feature can be used to hide files that are not to be executed for the time being without removing them. The command can be used only by the super-user.

Files in `/etc/rc2.d` must begin with an **S** or a **K** followed by a number and the rest of the file name. Upon entering run level 2, files beginning with **S** are executed with the **start** option; files beginning with **K**, are executed with the **stop** option. Files beginning with other characters are ignored.

SEE ALSO

`shutdown(1M)`.

NAME

relogin – rename login entry to show current layer

SYNOPSIS

`/usr/lib/layersys/relogin [-s] [line]`

DESCRIPTION

The *relogin* command changes the terminal *line* field of a user's *utmp*(4) entry to the name of the windowing terminal layer attached to standard input. *write*(1) messages sent to this user are directed to this layer. In addition, the *who*(1) command will show the user associated with this layer. *relogin* may only be invoked under *layers*(1).

relogin is invoked automatically by *layers*(1) to set the *utmp*(4) entry to the terminal line of the first layer created upon startup, and to reset the *utmp*(4) entry to the real line on termination. It may be invoked by a user to designate a different layer to receive *write*(1) messages.

-s Suppress error messages.

line Specifies which *utmp*(4) entry to change. The *utmp*(4) file is searched for an entry with the specified *line* field. That field is changed to the line associated with the standard input. (To learn what lines are associated with a given user, say *jd*oe, type `ps -f -u jd`oe and note the values shown in the TTY field (see *ps*(1))).

FILES

`/etc/utmp` database of users versus terminals

DIAGNOSTICS

Returns 0 upon successful completion, 1 otherwise.

SEE ALSO

layers(1), *mesg*(1), *ps*(1), *who*(1), *write*(1) in the *User's Reference Manual*.
utmp(4) in the *System Administrator's Reference Manual*.

NOTES

If *line* does not belong to the user issuing the *relogin* command or standard input is not associated with a terminal, *relogin* will fail.



NAME

rfadmin – Remote File Sharing administration

SYNOPSIS

rfadmin

rfadmin **[-[ar] domain.nodename**

rfadmin **[-[pq]**

rfadmin **-o option**

DESCRIPTION

rfadmin is primarily used to add and remove computers and their associated authentication information from a *domain/passwd* file on a Remote File Sharing primary domain name server. It is also used to transfer domain name server responsibilities from one machine to another. Used with no options, **rfadmin** returns the *domain.nodename* of the current domain name server for the local domain. Other options let you check if RFS is running and turn on the RFS loop back feature.

rfadmin can only be used to modify domain files on the primary domain name server (**-a** and **-r** options). If domain name server responsibilities are temporarily passed to a secondary domain name server, that computer can use the **-p** option to pass domain name server responsibility back to the primary. **rfadmin** can be used on any computer with no options or with the **q** or **o** options. to print information about the current domain name server. The user must have **root** permissions to use the command.

-a domain.nodename Used to add a computer to the member list of the domain that is served by this primary domain name server. The computer's name must be of the form *domain.nodename*. This command creates an entry for *nodename* in the *domain/passwd* file, which has the same format as */etc/passwd*, and prompts for an initial authentication password. The password prompting process conforms with that of **passwd(1)**.

-r domain.nodename Used to remove a computer from its domain by removing it from the *domain/passwd* file.

-p Used to pass the domain name server responsibilities back to a primary or to a secondary name server.

-q Prints a message that will tell you whether or not RFS is running.

-o option Lets you set RFS system options, by replacing *option* with one of the following:

loopback - Enables loop back facility for your computer. When this is set, you can mount a resource that is advertised from your own computer. This is used for testing applications in RFS when only one computer is available. Loop back is off by default.

noloopback - Turns off the loop back facility for your computer. This is the default.

ERRORS

When used with the **-a** option, if *domain.nodename* is not unique in the domain, an error message will be sent to standard error.

When used with the **-r** option, if (1) *domain.nodename* does not exist in the domain, (2) *domain.nodename* is defined as a domain name server, or (3) there are resources advertised by *domain.nodename*, an error message will be sent to standard error.

When used with the **-p** option to change the domain name server, if there are no backup name servers defined for *domain*, a warning message will be sent to standard error.

FILES

/usr/nserve/auth.info/domain/passwd

(For each *domain*, this file: is created on the primary, should be copied to all secondaries, and should be copied to all computers that want to do password verification of computers in the *domain*.)

SEE ALSO

passwd(1), *rfstart(1M)*, *rfstop(1M)*, *umount(1M)*.

NAME

`rfpasswd` – change Remote File Sharing host password

SYNOPSIS

`rfpasswd`

DESCRIPTION

`rfpasswd` updates the Remote File Sharing authentication password for a host; processing of the new password follows the same criteria as `passwd(1)`. The updated password is registered at the domain name server (`/usr/nserve/auth.info/domain/passwd`) and replaces the password stored at the local host (`/usr/nserve/loc.passwd` file).

This command is restricted to the super-user.

NOTE: If you change your host password, make sure that hosts that validate your password are notified of this change. To receive the new password, hosts must obtain a copy of the `domain/passwd` file from the domain's primary name server. If this is not done, attempts to mount remote resources may fail!

ERRORS

If (1) the old password entered from this command does not match the existing password for this machine, (2) the two new passwords entered from this command do not match, (3) the new password does not satisfy the security criteria in `passwd(1)`, (4) the domain name server does not know about this machine, or (5) the command is not run with super-user privileges, an error message will be sent to standard error. Also, Remote File Sharing must be running on your host and your domain's primary name server. A new password cannot be logged if a secondary is acting as the domain name server.

FILES

`/usr/nserve/auth.info/domain/passwd`
`/usr/nserve/loc.passwd`

SEE ALSO

`rfstart(1M)`, `rfadmin(1M)`.
`passwd(1)` in the *User's Reference Manual*.



NAME

rfstart – start Remote File Sharing

SYNOPSIS

rfstart [-v] [-p*primary_addr*]

DESCRIPTION

rfstart starts Remote File Sharing and defines an authentication level for incoming requests. (This command can only be used after the domain name server is set up and your computer's domain name and network specification has been defined using **dname**(1M).)

-v Specifies that verification of all clients is required in response to initial incoming mount requests; any host not in the file */usr/nserve/auth.info/domain/passwd* for the **domain** they belong to, will not be allowed to mount resources from your host. If **-v** is not specified, hosts named in *domain/passwd* will be verified, other hosts will be allowed to connect without verification.

-p primary_addr

Indicates the primary domain name server for your domain. *primary_addr* must be the network address of the primary name server for your domain. If the **-p** option is not specified, the address of the domain name server is taken from the **rfmaster**(4) file. (See **rfmaster**(4) for a description of the valid address syntax.)

If the host password has not been set, **rfstart** will prompt for a password; the password prompting process must match the password entered for your machine at the primary domain name server (see **rfadmin**(1M)). If you remove the *loc.passwd* file or change domains, you will also have to reenter the password.

Also, when **rfstart** is run on a domain name server, entries in the **rfmaster**(4) file are syntactically validated.

This command is restricted to the super-user.

ERRORS

If syntax errors are found in validating the **rfmaster**(4) file, a warning describing each error will be sent to standard error.

If (1) the shared resource environment is already running, (2) there is no communications network, (3) the domain name server cannot be found, (4) the domain name server does not recognize the machine, or (5) the command is run without super-user privileges, an error message will be sent to standard error.

Remote file sharing will not start if the host password in */usr/nserve/loc.passwd* is corrupted. If you suspect this has happened, remove the file and run **rfstart** again to reenter your password.

NOTE: **rfstart** will NOT fail if your host password does not match the password on the domain name server. You will simply receive a warning message. However, if you try to mount a resource from the primary or any other host that validates your password, the mount will fail if your password does not match the one that host has listed for your machine.

FILES

/usr/nserve/rfmaster
/usr/nserve/loc.passwd

SEE ALSO

adv(1M), dname(1M), mount(1M), rfadmin(1M), rfstop(1M), unadv(1M), rfmaster(4).

NAME

rfstop – stop the Remote File Sharing environment

SYNOPSIS

rfstop

DESCRIPTION

rfstop disconnects a host from the Remote File Sharing environment until another *rfstart(1M)* is executed.

When executed on the domain name server, the domain name server responsibility is moved to a secondary name server as designated in the *rfmaster(4)* file. If there is no designated secondary name server **rfstop** will issue a warning message, Remote File Sharing will be stopped, and name service will no longer be available to the domain.

This command is restricted to the super-user.

ERRORS

If (1) there are resources currently advertised by this host, (2) resources from this machine are still remotely mounted by other hosts, (3) there are still remotely mounted resources in the local file system tree, (4) *rfstart(1M)* had not previously been executed, or (5) the command is not run with super-user privileges, an error message will be sent to standard error and Remote File Sharing will not be stopped.

SEE ALSO

adv(1M), *mount(1M)*, *rfadmin(1M)*, *rfstart(1M)*, *unadv(1M)*, *rfmaster(4)*.



NAME

rfuadmin – Remote File Sharing notification shell script

SYNOPSIS

rfuadmin *message remote_resource* [*seconds*]

DESCRIPTION

The **rfuadmin** administrative shell script responds to unexpected Remote File Sharing events, such as broken network connections and forced unmounts, picked up by the **rfudaemon** process. This command is not intended to be run directly from the shell.

The response to messages received by **rfudaemon** can be tailored to suit the particular system by editing the **rfuadmin** script. The following paragraphs describe the arguments passed to **rfuadmin** and the responses.

disconnect *remote_resource*

A link to a remote resource has been cut. **rfudaemon** executes **rfuadmin**, passing it the message **disconnect** and the name of the disconnected resource. **rfuadmin** sends this message to all terminals using **wall(1)**:

Remote_resource has been disconnected from the system.

Then it executes **fuser(1M)** to kill all processes using the resource, unmounts the resource [**umount(1M)**] to clean up the kernel, and starts **rmount** to try to remount the resource.

fumount *remote_resource*

A remote server machine has forced an unmount of a resource a local machine has mounted. The processing is similar to processing for a disconnect.

fuwarn *remote_resource seconds*

This message notifies **rfuadmin** that a resource is about to be unmounted. **rfudaemon** sends this script the **fuwarn** message, the resource name, and the number of seconds in which the forced unmount will occur. **rfuadmin** sends this message to all terminals:

Remote_resource is being removed from the system in # seconds.

SEE ALSO

fumount(1M), **rmount(1M)**, **rfudaemon(1M)**, **rfstart(1M)**.
wall(1) in the *User's Reference Manual*.

BUGS

The console must be on when Remote File Sharing is running. If it's not, **rfuadmin** will hang when it tries to write to the console (**wall**) and recovery from disconnected resources will not complete.



NAME

rfudaemon – Remote File Sharing daemon process

SYNOPSIS

rfudaemon

DESCRIPTION

The **rfudaemon** command is started automatically by **rfstart(1M)** and runs as a daemon process as long as Remote File Sharing is active. Its function is to listen for unexpected events, such as broken network connections and forced unmounts, and execute appropriate administrative procedures.

When such an event occurs, **rfudaemon** executes the administrative shell script **rfuadmin**, with arguments that identify the event. This command is not intended to be run from the shell. Here are the events:

DISCONNECT

A link to a remote resource has been cut. **rfudaemon** executes **rfuadmin**, with two arguments: **disconnect** and the name of the disconnected resource.

FUMOUNT

A remote server machine has forced an unmount of a resource a local machine has mounted. **rfudaemon** executes **rfuadmin**, with two arguments: **fumount** and the name of the disconnected resource.

GETUMSG

A remote user-level program has sent a message to the local **rfudaemon**. Currently the only message sent is *fuwarn*, which notifies **rfuadmin** that a resource is about to be unmounted. It sends **rfuadmin** the *fuwarn*, the resource name, and the number of seconds in which the forced unmount will occur.

LASTUMSG

The local machine wants to stop the **rfudaemon** [**rfstop(1M)**]. This causes **rfudaemon** to exit.

SEE ALSO

rfstart(1M), **rfuadmin(1M)**.



NAME

`rmntstat` – display mounted resource information

SYNOPSIS

`rmntstat [-h] [resource]`

DESCRIPTION

When used with no options, *rmntstat* displays a list of all local Remote File Sharing resources that are remotely mounted, the local path name, and the corresponding clients. *rmntstat* returns the remote mount data regardless of whether a resource is currently advertised; this ensures that resources that have been unadvertised but are still remotely mounted are included in the report. When a *resource* is specified, *rmntstat* displays the remote mount information only for that resource. The *-h* option causes header information to be omitted from the display.

EXIT STATUS

If no local resources are remotely mounted, *rmntstat* will return a successful exit status.

ERRORS

If *resource* (1) does not physically reside on the local machine or (2) is an invalid resource name, an error message will be sent to standard error.

SEE ALSO

`mount(1M)`, `fumount(1M)`, `unadv(1M)`.



NAME

`rmnttry` – attempt to mount queued remote resources

SYNOPSIS

`/usr/nserve/rmnttry` [*resource ...*]

DESCRIPTION

`rmnttry` sequences through the pending mount requests stored in `/usr/nserve/rmnttab`, trying to mount each resource. If a mount succeeds, the resource entry is removed from the `/usr/nserve/rmnttab` file.

If one or more resource names are supplied, mounts are attempted only for those resources, rather than for all pending mounts. Mounts are not attempted for resources not present in the `/usr/nserve/rmnttab` file (see `rmount(1M)`). If a mount invoked from `rmnttry` takes over 3 minutes to complete, `rmnttry` aborts the mount and issues a warning message.

`rmnttry` is typically invoked from a cron entry in `/usr/spool/cron/crontabs/root` to attempt mounting queued resources at periodic intervals. The default strategy is to attempt mounts at 15 minute intervals. The cron entry for this is:

```
10,25,40,55 * * * * /usr/nserve/rmnttry
>/dev/null
```

FILES

`/usr/nserve/rmnttab` pending mount requests

SEE ALSO

`mount(1M)`, `rmount(1M)`, `rumount(1M)`, `mnttab(4)`.
`crontab(1)` in the *User's Reference Manual*.

DIAGNOSTICS

An exit code of 0 is returned if all requested mounts succeeded, 1 is returned if one or more mounts failed, and 2 is returned for bad usage.



NAME

rmount – queue remote resource mounts

SYNOPSIS

/etc/rmount [-d[r] *resource directory*]

DESCRIPTION

rmount queues a remote resource for mounting. The command enters the resource request into **/usr/nserve/rmnttab**, which is formatted identically to **mnttab(4)**. **rmntry(1M)** is used to poll entries in this file.

When used without arguments, **rmount** prints a list of resources with pending mounts along with their destined directories, modes, and date of request. The resources are listed chronologically, with the oldest resource request appearing first.

The following options are available:

- d indicates that the *resource* is a remote resource to be mounted on *directory*.
- r indicates that the *resource* is to be mounted read-only. If the *resource* is write-protected, this flag must be used.

FILES

/usr/nserve/rmnttab pending mount requests

SEE ALSO

mount(1M), **rmntry(1M)**, **rumount(1M)**, **rmountall(1M)**, **mnttab(4)**.

DIAGNOSTICS

An exit code of 0 is returned upon successful completion of **rmount**. Otherwise, a non-zero value is returned.



NAME

rmountall, rumountall – mount, unmount Remote File Sharing resources

SYNOPSIS

```
/etc/rmountall [-] " file-system-table " [...]
/etc/rumountall [ -k ]
```

DESCRIPTION

rmountall is a Remote File Sharing command used to mount remote resources according to a *file-system-table*. (*/etc/fstab* is the recommended *file-system-table*.) **rmountall** also invokes the */usr/nserve/rmnttry(1M)* command which attempts to mount queued resources. The special file name "-" reads from the standard input.

rumountall causes all mounted remote resources to be unmounted and deletes all resources that were queued from */etc/rmount(1M)*. The **-k** option sends a SIGKILL signal, via *fuser(1M)*, to processes that have files open.

These commands may be executed only by the super-user.

The file-system-table format is as follows:

column 1	block special file name of file system
column 2	mount-point directory
column 3	-r if to be mounted read-only; -d if remote resource
column 4	file system type (not used with Remote File Sharing)
column 5+	ignored

White-space separates columns. Lines beginning with "#" are comments. Empty lines are ignored.

SEE ALSO

fuser(1M), *mount(1M)*, *rfstart(1M)*, *rmnttry(1M)*, *rmount(1M)*, *sysadm(1)* in the *User's Reference Manual*.
signal(2) in the *Programmer's Reference Manual*.

DIAGNOSTICS

No messages are printed if the remote resources are mounted successfully.

Error and warning messages come from *mount(1M)*.



NAME

`rumount` – cancel queued remote resource request

SYNOPSIS

`/usr/nserve/rumount resource ...`

DESCRIPTION

`rumount` cancels a request for one or more resources that are queued for mount. The entries for the resources are deleted from `/usr/nserve/rmnttab`.

FILES

`/usr/nserve/rmnttab` pending mount requests

SEE ALSO

`mount(1M)`, `rmntry(1M)`, `rmount(1M)`, `rumountall(1M)`, `mnttab(4)`.

DIAGNOSTICS

An exit code of 0 is returned if `rumount` completes successfully. A 1 is returned if the resource requested for dequeuing is not in `/usr/nserve/rmnttab`, and a 2 is returned for bad usage or an error in reading or writing `/usr/nserve/rmnttab`.



NAME

sadp – disk access profiler

SYNOPSIS

sadp [*-th*] [*-d* device[*-drive*]] s [*n*]

DESCRIPTION

sadp reports disk access location and seek distance, in tabular or histogram form. It samples disk activity once every second during an interval of *s* seconds. This is done *n* times if *n* is specified. Cylinder usage and disk distance are recorded in units of 8 cylinders.

Valid values of *device* are **hdsk** for integral disk, **sdk** for the Small Computer Systems Interface (SCSI) disk, and **fdsk** for integral floppy. Neither XDC disks nor SCSI Release 1.0 disks can be profiled using *sadp*. *sadp* can profile only one device type per invocation. The *-d* option may be omitted if the system has only one device type.

Drive specifies the disk drives and it may be:

a drive number in the range supported by *device*,
two numbers separated by a minus (indicating an inclusive range),

or

a list of drive numbers separated by commas.

Up to 8 disk drives may be reported for device type **hdsk** or **fdsk**, and up to 56 for **sdk**. If *drive* is not specified, *sadp* profiles all the disk drives specified by *device*, up to the maximum of 8 for **hdsk** and **fdsk**, or 56 for **sdk**.

The *-t* flag causes the data to be reported in tabular form. The *-h* flag produces a histogram of the data. Default is *-t*.

EXAMPLE

The command:

```
sadp -d hdsk-0 900 4
```

will generate 4 tabular reports, each describing cylinder usage and seek distance of **hdsk** disk drive 0 during a 15-minute interval.

FILES

/dev/kmem

SEE ALSO

mem(7).



NAME

sar: sa1, sa2, sadc – system activity report package

SYNOPSIS

```
/usr/lib/sa/sadc [t n] [ofile]
```

```
/usr/lib/sa/sa1 [t n]
```

```
/usr/lib/sa/sa2 [-ubdycwaqvmprDSAC] [-s time] [-e time] [-i sec]
```

DESCRIPTION

System activity data can be accessed at the special request of a user (see *sar(1)*) and automatically on a routine basis as described here. The operating system contains several counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, paging and Remote File Sharing.

sadc and shell procedures, *sa1* and *sa2*, are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of *sadc* itself may affect the sample. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time, when booting to a multiuser state, to mark the time at which the counters restart from zero. For example, the `/etc/init.d/perf` file writes the restart mark to the daily data by the command entry:

```
su sys -c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

The shell script *sa1*, a variant of *sadc*, is used to collect and store data in binary file `/usr/adm/sa/sadd` where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The entries in `/usr/spool/cron/crontabs/sys` (see *cron(1M)*):

```
0 * * 0-6 /usr/lib/sa/sa1
20,40 8-17 * * 1-5 /usr/lib/sa/sa1
```

will produce records every 20 minutes during working hours and hourly otherwise.

The shell script *sa2*, a variant of *sar(1)*, writes a daily report in file `/usr/adm/sa/sardd`. The options are explained in *sar(1)*. The `/usr/spool/cron/crontabs/sys` entry:

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

will report important activities hourly during the working day.

The structure of the binary daily data file is:

```

struct sa {
    struct sysinfo si; /* see /usr/include/sys/sysinfo.h */
    struct minfo mi; /* defined in sys/sysinfo.h */
    struct dinfo di; /* RFS info defined in sys/sysinfo.h */
    struct rcinfo rc; /* Client cache info defined in sys/sysinfo.h */
    struct bpbinfo bi; /* Co-processor info defined in sys/sysinfo.h */
    int bpb_utilize /* Co-processor utilize flag */
    int minserve, maxserve; /* RFS server low and high water marks */
    int szinode; /* current size of inode table */
    int szfile; /* current size of file table */
    int szproc; /* current size of proc table */
    int szlckf; /* current size of file record header table */
    int szlckr; /* current size of file record lock table */
    int mszinode; /* size of inode table */
    int mszfile; /* size of file table */
    int mszproc; /* size of proc table */
    int mszlckf; /* maximum size of file record header table */
    int mszlckr; /* maximum size of file record lock table */
    long inodeovf; /* cumulative overflows of inode table */
    long fileovf; /* cumulative overflows of file table */
    long procovf; /* cumulative overflows of proc table */
    time_t ts; /* time stamp, seconds */
    long devio[NDEVS][4]; /* device unit information */
#define IO_OPS 0 /* cumulative I/O requests */
#define IO_BCNT 1 /* cumulative blocks transferred */
#define IO_ACT 2 /* cumulative drive busy time in ticks */
#define IO_RESP 3 /* cumulative I/O resp time in ticks */
};

```

FILES

/usr/adm/sa/sadd	daily data file
/usr/adm/sa/sardd	daily report file
/tmp/sa.adrfl	address file

SEE ALSO

cron(1M), sag(1G), sar(1), timex(1).

NAME

setclk – set system time from hardware clock

SYNOPSIS

/etc/setclk

DESCRIPTION

setclk is used to set the internal system time from the hardware time-of-day clock. The command can be used only by the super-user. It is normally executed by an entry in the */etc/inittab* file when the system is initialized at boot time. Note that *setclk* checks the Nonvolatile Random Access Memory (NVRAM) only for the date. If the date is set, *setclk* runs silently. If the date is not set, *setclk* prompts the user to use *sysadm datetime* (see *sysadm[1]*) for the proper setting of the hardware clock.

SEE ALSO

sysadm(1) in the *User's Reference Manual*.



NAME

setmnt – establish mount table

SYNOPSIS

/etc/setmnt

DESCRIPTION

setmnt creates the */etc/mnttab* table which is needed for both the *mount*(1M) and *umount* commands. *setmnt* reads standard input and creates a *mnttab* entry for each line. Input lines have the format:

fileSYS node

where *fileSYS* is the name of the file system's *special file* (e.g., */dev/dsk/c?d?s?*) and *node* is the root name of that file system. Thus *fileSYS* and *node* become the first two strings in the mount table entry.

FILES

/etc/mnttab

SEE ALSO

mount(1M).

BUGS

Problems may occur if *fileSYS* or *node* are longer than 32 characters.

setmnt silently enforces an upper limit on the maximum number of *mnttab* entries.



NAME

shutdown – shut down system, change system state

SYNOPSIS

`/etc/shutdown [-y] [-ggrace_period [-iinit_state]`

DESCRIPTION

This command is executed by the super-user to change the state of the machine. By default, it brings the system to a state where only the console has access to the UNIX system. This state is traditionally called "single-user".

The command sends a warning message and a final message before it starts actual shutdown activities. By default, the command asks for confirmation before it starts shutting down daemons and killing processes. The options are used as follows:

-y pre-answers the confirmation question so the command can be run without user intervention. A default of 60 seconds is allowed between the warning message and the final message. Another 60 seconds is allowed between the final message and the confirmation.

-ggrace_period
allows the super-user to change the number of seconds from the 60-second default.

-iinit_state
specifies the state that *init*(1M) is to be put in following the warnings, if any. By default, system state "s" is used (the same as states "1" and "S").

Other recommended system state definitions are:

state 0

Shut the machine down so it is safe to remove the power. Have the machine remove power if it can. The */etc/rc0* procedure is called to do this work.

state 1, s, S

Bring the machine to the state traditionally called single-user. The */etc/rc0* procedure is called to do this work. (Though s and 1 are both used to go to single user state, s only kills processes spawned by *init* and does not unmount file systems. State 1 unmounts everything except root and kills all user processes, except those that relate to the console.)

state 5

Stop the UNIX system and go to the firmware monitor.

state 6

Stop the UNIX system and reboot to the state defined by the *initdefault* entry in */etc/inittab*.

SEE ALSO

init(1M), *rc0*(1M), *rc2*(1M), *inittab*(4).



NAME

strace – print STREAMS trace messages

SYNOPSIS

strace [*mid sid level*] ...

DESCRIPTION

strace without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver [*log(7)*]. If arguments are provided they must be in triplets of the form *mid, sid, level*, where *mid* is a STREAMS module id number, *sid* is a sub-id number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module/driver, sub-id (usually indicating minor device), and priority level equal to or less than the given level. The token *all* may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is:

```
<seq> <time> <ticks> <level> <flags> <mid> <sid> <text>
  <seq>   trace sequence number
  <time>   time of message in hh:mm:ss
  <ticks>  time of message in machine ticks since boot
  <level>  tracing priority level
  <flags>  E : message is also in the error log
           F : indicates a fatal error
           N : mail was sent to the system administrator
  <mid>    module id number of source
  <sid>    sub-id number of source
  <text>   formatted text of the trace message
```

Once initiated, *strace* will continue to execute until terminated by the user.

EXAMPLES

Output all trace messages from the module or driver whose module id is 41:

```
strace 41 all all
```

Output those trace messages from driver/module id 41 with sub-ids 0, 1, or 2:

```
strace 41 0 1 41 1 1 41 2 0
```

Messages from sub-ids 0 and 1 must have a tracing level less than or equal to 1. Those from sub-id 2 must have a tracing level of 0.

CAVEATS

Due to performance considerations, only one *strace* process is permitted to open the STREAMS log driver at a time. The log driver has a list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the *strace* process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running *strace* will have the most impact on the timing of the modules and drivers generating the trace messages that are

sent to the *strace* process. If trace messages are generated faster than the *strace* process can handle them, then some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

SEE ALSO

log(7).

STREAMS Programmer's Guide.

NAME

`strclean` – STREAMS error logger cleanup program

SYNOPSIS

`strclean [-d logdir] [-a age]`

DESCRIPTION

`strclean` is used to clean up the STREAMS error logger directory on a regular basis (for example, by using `cron(1M)`). By default, all files with names matching `error.*` in `/usr/adm/streams` that have not been modified in the last 3 days are removed. A directory other than `/usr/adm/streams` can be specified using the `-d` option. The maximum age in days for a log file can be changed using the `-a` option.

EXAMPLE

```
strclean -d /usr/adm/streams -a 3
```

has the same result as running `strclean` with no arguments.

NOTES

`strclean` is typically run from `cron(1M)` on a daily or weekly basis.

FILES

`/usr/adm/streams/error.*`

SEE ALSO

`cron(1M)`, `strerr(1M)`.
STREAMS Programmer's Guide.



NAME

strerr – STREAMS error logger daemon

SYNOPSIS

strerr

DESCRIPTION

strerr receives error log messages from the STREAMS log driver [*log(7)*] and appends them to a log file. The error log files produced reside in the directory */usr/adm/streams*, and are named **error.mm-dd**, where *mm* is the month and *dd* is the day of the messages contained in each log file.

The format of an error log message is:

```
<seq> <time> <ticks> <flags> <mid> <sid> <text>
  <seq>   error sequence number
  <time>  time of message in hh:mm:ss
  <ticks>  time of message in machine ticks since boot priority level
  <flags>  T : the message was also sent to a tracing process
           F : indicates a fatal error
           N : send mail to the system administrator
  <mid>   module id number of source
  <sid>   sub-id number of source
  <text>  formatted text of the error message
```

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages which indicate the total failure of a STREAMS driver or module should have the F flag set. Those messages requiring the immediate attention of the administrator will have the N flag set, which causes the error logger to send the message to the system administrator via *mail(1)*. The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, *strerr* will continue to execute until terminated by the user. Commonly, *strerr* would be executed asynchronously.

CAVEATS

Only one *strerr* process at a time is permitted to open the STREAMS log driver.

If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

FILES

/usr/adm/streams/error.mm-dd

SEE ALSO

log(7).
STREAMS Programmer's Guide.



NAME

`su` – become super-user or another user

SYNOPSIS

`su [-] [name [arg ...]]`

DESCRIPTION

`su` allows one to become another user without logging off. The default user *name* is **root** (i.e., super-user).

To use `su`, the appropriate password must be supplied (unless one is already **root**). If the password is correct, `su` will execute a new shell with the real and effective user ID set to that of the specified user. The new shell will be the optional program named in the shell field of the specified user's password file entry (see `passwd(4)`), or `/bin/sh` if none is specified (see `sh(1)`). To restore normal user ID privileges, type an EOF (`cntrl-d`) to the new shell.

Any additional arguments given on the command line are passed to the program invoked as the shell. When using programs like `sh(1)`, an *arg* of the form `-c string` executes *string* via the shell and an *arg* of `-r` will give the user a restricted shell.

The following statements are true only if the optional program named in the shell field of the specified user's password file entry is like `sh(1)`. If the first argument to `su` is a `-`, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is `-`, thus causing first the system's profile (`/etc/profile`) and then the specified user's profile (`.profile` in the new HOME directory) to be executed. Otherwise, the environment is passed along with the possible exception of `$PATH`, which is set to `/bin:/etc:/usr/bin` for **root**. Note that if the optional program used as the shell is `/bin/sh`, the user's `.profile` can check *arg0* for `-sh` or `-su` to determine if it was invoked by `login(1)` or `su(1)`, respectively. If the user's program is other than `/bin/sh`, then `.profile` is invoked with an *arg0* of `-program` by both `login(1)` and `su(1)`.

All attempts to become another user using `su` are logged in the log file `/usr/adm/sulog`.

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute:

```
su bin
```

To become user **bin** but change the environment to what would be expected if **bin** had originally logged in, execute:

```
su - bin
```

To execute *command* with the temporary environment and permissions of user **bin**, type:

```
su - bin -c "command args"
```

FILES

/etc/passwd	system's password file
/etc/profile	system's profile
\$HOME/.profile	user's profile
/usr/adm/sulog	log file

SEE ALSO

env(1), login(1), sh(1) in the *User's Reference Manual*.
passwd(4), profile(4), environ(5) in the *System Administrator's Reference Manual*.

NAME

swap – swap administrative interface

SYNOPSIS

```
/etc/swap -a swapdev swaplow swaplen
/etc/swap -d swapdev swaplow
/etc/swap -l
```

DESCRIPTION

swap provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager. The following options are recognized:

- a Add the specified swap area. *swapdev* is the name of the block special device, e.g., */dev/dsk/1s0*. *swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. *swaplen* is the length of the swap area in 512-byte blocks. This option can only be used by the super-user. Swap areas are normally added by the system start up routine */etc/rc* when going into multi-user mode.
- d Delete the specified swap area. *swapdev* is the name of block special device, e.g., */dev/dsk/1s0*. *swaplow* is the offset in 512-byte blocks into the device where the swap area should begin. Using this option marks the swap area as "INDEL" (in process of being deleted). The system will not allocate any new blocks from the area, and will try to free swap blocks from it. The area will remain in use until all blocks from it are freed. This option can only be used by the super-user.
- l List the status of all the swap areas. The output has four columns:

DEV	The <i>swapdev</i> special file for the swap area if one can be found in the <i>/dev/dsk</i> or <i>/dev</i> directories, and its major/minor device number in decimal.
LOW	The <i>swaplow</i> value for the area in 512-byte blocks.
LEN	The <i>swaplen</i> value for the area in 512-byte blocks.
FREE	The number of free 512-byte blocks in the area. If the swap area is being deleted, this column will be marked INDEL.

WARNINGS

No check is done to see if a swap area being added overlaps with an existing swap area or file system.



NAME

`sync` -- update the super block

SYNOPSIS

`sync`

DESCRIPTION

`sync` executes the `sync` system primitive. If the system is to be stopped, `sync` must be called to insure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See `sync(2)` for details.

NOTE

If you have done a write to a file on a remote machine in a Remote File Sharing environment, you cannot use `sync` to force buffers to be written out to disk on the remote machine. `sync` will only write local buffers to local disks.

SEE ALSO

`sync(2)` in the *Programmer's Reference Manual*.



NAME

`sysadm` – menu interface to do system administration

SYNOPSIS

`sysadm` [*sub-command*]

DESCRIPTION

This command, when invoked without an argument, presents a menu of system administration sub-commands, from which the user selects. If the optional argument is presented, the named sub-command is run or the named sub-menu is presented.

The `sysadm` command may be given a password. See `admpasswd` in the SUB-COMMANDS section.

SUB-COMMANDS

The following menus of sub-commands are available. (The number of bullets (•) in front of each item indicates the level of the menu or subcommand.)

- diagnostics
system diagnostics menu

These subcommands look for and sometimes repair problems in the system. Those subcommands that issue reports allow you to determine if there are detectable problems. Commands that attempt repair are for repair people only. You must know what you are doing!

- diskrepair
advice on repair of built-in disk errors

This subcommand advises you on how to go about repairing errors that occur on built-in disks.

WARNING: Because this is a repair function, it should only be performed by qualified service personnel.

NOTE: Reports of disk errors most probably result in the loss of files and/or damage to data. It will be necessary to restore the repaired disk from backup copies.

- diskreport
report on built-in disk errors

This subcommand shows you if the system has collected any information indicating that there have been errors while reading the built-in disks. You can request either summary or full reports. The summary report provides sufficient information about disk errors to determine if repair should be attempted. If the message no errors logged is part of the report, then there is probably no damage. If a number of errors is reported, there is damage and you should call for service. The full report gives additional detail for the expert repair person trouble shooting complicated problems.

NOTE: Reports of disk errors most probably result in the loss of files and/or damage to data. It will be necessary to restore the repaired disk from backup copies.

- diskmgmt

disk management menu

The subcommands in this menu provide functions for using removable disks. The subcommands include the ability to format disks, copy disks, and to use disks as mountable file systems. It also contains a menu of subcommands for handling non-removable media.

- checkfsys

check a removable disk file system for errors

Checkfsys checks a file system on a removable disk for errors. If there are errors, this procedure attempts to repair them.

- cpdisk

make exact copies of a removable disk

This procedure copies the contents of a removable disk into the machine and then allows the user to make exact copies of it. These copies are identical to the original in every way. The copies are made by first reading the original removable disk entirely into the machine and then writing it out onto duplicate disks. The procedure will fail if there is not enough space in the system to hold the original disk.

- erase

erase data from removable disk

This procedure erases a removable disk by overwriting it with null bytes. The main purpose is to remove data that the user does not want seen. Once performed, this operation is irreversible.

- format

format new removable disks

Format prepares new removable disks for use. Once formatted, programs and data can be written on the disks.

- harddisk

hard disk management menu

The subcommands in this menu provide functions for using hard disks. For each hard disk, the disk can be partitioned with default partitioning or the current disk partitioning can be displayed.

- display

display hard disk partitioning

Display will allow the user to display the hard disk partitioning. This will inform the user of current disk partitioning information.

- partitioning
 - partition a hard disk

Partitioning configures hard disks. This will allow you to partition a hard disk according to the default partitioning.

- rmdisk
 - remove a hard disk

Removes a hard disk from the system configuration. It may then be physically disconnected (once the machine has been turned off) or freshly partitioned (after the machine has been restarted).

- makefsys
 - create a new file system on a removable disk

Makefsys creates a new file system on a removable disk which can then store data which the user does not wish to keep on the hard disk. When "mounted", the file system has all the properties of a file kept on the hard disk, except that it is smaller.

- mountfsys
 - mount a removable disk file system

Mountfsys mounts a file system, found on a removable disk, making it available to the user. The file system is unmounted with the "umountfsys" command. **THE DISK MUST NOT BE REMOVED WHILE THE FILE SYSTEM IS STILL MOUNTED. IF THE FILE SYSTEM HAS BEEN MOUNTED WITH THE mountfsys COMMAND, IT MUST BE UNMOUNTED WITH umountfsys.**

- umountfsys
 - unmount a removable disk file system

Umountfsys unmounts a file system, allowing the user to remove the disk. **THE DISK MUST NOT BE REMOVED UNTIL THE FILE SYSTEM IS UNMOUNTED. umountfsys MAY ONLY BE USED TO UNMOUNT FILE SYSTEMS MOUNTED WITH THE mountfsys COMMAND.**

- filemgmt
 - file management menu

The subcommands in this menu allow the user to protect files on the hard disk file systems by copying them onto diskettes and later restoring them to the hard disk by copying them back. Subcommands are also provided to determine which files might be best kept on diskette based on age or size.

- backup

backup files from integral hard disk to removable disk or tape

Backup saves copies of files from the integral hard disk file systems to removable disk or tape. There are two kinds of backups:

COMPLETE – copies all files (useful in case of serious file system damage)

INCREMENTAL – copies files changed since the last backup

The normal usage is to do a complete backup of each file system and then periodically do incremental backups. Two cycles are recommended (one set of complete backups and several incrementals to each cycle). Files backed up with "backup" are restored using "restore".

- bupsched

backup reminder scheduling menu

Backup scheduling is used to schedule backup reminder messages and backup reminder checks. Backup reminder messages are sent to the console to remind the administrator to backup particular file systems when the machine is shutdown or a reminder check has been run during the specified time period.

Backup reminder checks specify particular times at which the system will check to see if any backup reminder messages have been scheduled.

- schedcheck

schedule backup reminder checks

Backup reminder checks are run at specific times to check to see if any reminders are scheduled. The user specifies the times at which the check is to be run. Checks are run for the reminder messages scheduled by *schdmsg*.

- schdmsg

schedule backup reminder message

Backup reminder messages are sent to the console if the machine is shutdown or a reminder check has been scheduled. The user specifies the times at which it is appropriate to send a message and the file systems to be included in the message.

- diskuse

display how much of the hard disk is being used

Diskuse lets the user know what percentage of the hard disk is currently occupied by files. The list is organized by file system names.

- fileage

list files older than a particular date

Fileage prints the names of all files older than the date specified by the user. If no date is entered, all files older than 90 days will be listed.

- filesize

list the largest files in a particular directory

Filesize prints the names of the largest files in a specific directory. If no directory is specified, the `/usr/admin` directory will be used. If the user does not specify how many large files to list, 10 files will be listed.

- restore

restore files from "backup" and "store" media to integral hard disk

Restore copies files from disks and tapes made by "backup" and "store" back onto the hard disk. You can restore individual files, directories of files, or the entire contents of a disk or tape. The user can restore from both "incremental" and "complete" media. The user can also list the names of files stored on the disk or tape.

- store

store files and directories of files onto disk or tape

Store copies files from the integral hard disk to disk or tape and allows the user to optionally verify that they worked and to optionally remove them when done. Typically, these would be files that the user wants to archive or restrict access to. The user can store single files and directories of files. Use the "restore" command to put stored files back on the integral hard disk and to list the files stored.

- machinemgmt

machine management menu

Machine management functions are tools used to operate the machine, e.g., turn it off, reboot, or go to the firmware monitor.

- autold

set automatic boot device, default manual boot program

This procedure specifies the default manual program to boot from firmware and/or the device to be used when automatically rebooting.

- firmware

stop all running programs then enter firmware mode

This procedure will stop all running programs, close any open files, write out information to the disk (such as directory information), then enter the firmware mode. (Machine diagnostics and other special functions that are not available on the UNIX system.)

- floppykey

create a "floppy key" removable disk

The "floppy key" removable disk allows the user to enter firmware mode if the firmware password has been changed and then forgotten. Thus the "floppy key" is just that, the "key" to the system and should be protected as such.

- powerdown

stop all running programs, then turn off the machine

Powerdown will stop all running programs, close any open files, write out information to disk (such as directory information), then turn the machine power off.

- reboot

stop all running programs then reboot the machine

Reboot will stop all running programs, close any open files, write out information to disk (such as directory information), then reboot the machine. This can be used to get out of some types of system trouble, such as when a process cannot be killed.

- whoson

print list of users currently logged onto the system

Whoson prints the login ID, terminal device number, and sign-on time of all users who are currently using the computer.

- packagemgmt

package management

These submenus and subcommands manage various software and hardware packages that you install on your machine. Not all optional packages add subcommands here.

- **softwaremgmt**
software management menu

These subcommands permit the user to install new software, remove software, and run software directly from the removable disk it is delivered on. The "remove" and "run" capabilities are dependent on the particular software packages. See the instructions delivered with each package.

- **installpkg**
install new software package onto integral hard disk

Install copies files from removable disk onto the integral hard disk and performs additional work if necessary so that the software can be run. From then on, the user will have access to those commands.

- **listpkg**
list packages already installed

This subcommand shows you a list of currently installed optional software packages.

- **removepkg**
remove previously installed package from integral hard disk

This subcommand displays a list of currently installed optional software packages. Actions necessary to remove the software packages specified by the user will then be performed. The removable disk used to "installpkg" the software is needed to remove it.

- **runpkg**
run software package without installing it

This package allows the user to run software from a removable disk without installing it permanently on the system. This is useful if the user does not use the software often or does not have enough room on the system. **WARNING:** Not all software packages have the ability to run their contents this way. See the instructions that come with the software package.

- **syssetup**
system setup menu

System setup routines allow the user to tell the computer what its environment looks like: what the date, time, and time zone is, what administration and system capabilities are to be under password control, what the machine's name is, etc. The first-time setup sequence is also here.

- **admpasswd**
assign or change administrative passwords

Admpasswd lets you set or make changes to passwords for administrative commands and logins such as setup and sysadm.

- **datetime**
set the date, time, time zone, and daylight savings time

Datetime tells the computer the date, time, time zone, and whether you observe Daylight Savings Time (DST). It is normally run once when the machine is first set up. If you observe DST, the computer will automatically start to observe it in the spring and return to Standard Time in the fall. The machine has to be turned off and turned back on again to guarantee that ALL times will be reported correctly. Most are correct the next time the user logs in.

- **nodename**
set the node name of this machine

This allows you to change the node name of this machine. The node name is used by various communications networks to identify this machine.

- **setup**
set up your machine the very first time

Setup allows the user to define the first login, to set the passwords on the user-definable administration logins and to set the time zone for your location.

- **sypasswd**
assign system passwords

Sypasswd lets the user set system passwords normally reserved for the very knowledgeable user. For this reason, this procedure may assign those passwords, but may not change or clear them. Once set, they may only be changed by the specific login or the "root" login.

- **ttymgmt**
terminal management

This procedure allows the user to manage the computer's terminal functions.

- **lineset**
show tty line settings and hunt sequences

The tty line settings are often hunt sequences where, if the first line setting does not work, the line "hunts" to the next line setting until one that does work comes by. This subcommand shows the various sequences with only specific line settings in them. It also shows each line setting in detail.

- **mklineset**
create new tty line settings and hunt sequences

This subcommand helps you to create tty line setting entries. You might want to add line settings that are not in the current set or create hunt sequences with only specific line settings in them. The created hunt sequences are circular; stepping past the last setting puts you on the first.

- **modtty**
show and optionally modify characteristics of tty lines

This subcommand reports and allows you to change the characteristics of tty lines (also called "ports").

- **usermgmt**
user management menu

These subcommands allow you to add, modify and delete the list of users that have access to your machine. You can also place them in separate groups so that they can share access to files within the group but protect themselves from other groups.

- **addgroup**
add a group to the system

Addgroup adds a new group name or ID to the computer. Group names and IDs are used to identify groups of users who desire common access to a set of files and directories.

- **adduser**
add a user to the system

Adduser installs a new login ID on the machine. You are asked a series of questions about the user and then the new entry is made. You can enter more than one user at a time. Once this procedure is finished, the new login ID is available.

- **delgroup**
delete a group from the system

Delgroup allows you to remove groups from the computer. The deleted group is no longer identified by name. However, files may still be identified with the group ID number.

- **deluser**
delete a user from the system

Deluser allows you to remove users from the computer. The deleted user's files are removed from the hard disk and their logins are removed from the `/etc/passwd` file.

- **lsgroup**
list groups in the system

Lsgroup will list all the groups that have been entered into the computer. This list is updated automatically by "addgroup" and "delgroup".
- **lsuser**
list users in the system

Lsuser will list all the users that have been entered into the computer. This list is updated automatically by "adduser" and "deluser".
- **modadduser**
modify defaults used by adduser

Modadduser allows the user to change some of the defaults used when adduser creates a new login. Changing the defaults does not effect any existing logins, only logins made from this point on.
- **modgroup**
make changes to a group on the system

Modgroup allows the user to change the name of a group that the user enters when "addgroup" is run to set up new groups.
- **moduser**
menu of commands to modify a user's login

This menu contains commands that modify the various aspects of a user's login.
- **chgloginid**
change a user's login ID

This procedure allows the user to change a user's login ID. Administrative and system logins cannot be changed.
- **chgpasswd**
change a user's passwd

This procedure allows removal or change of a user's password. Administrative and system login passwords cannot be changed. To change administrative and system login passwords, see the system setup menu: sysadm syssetup.
- **chgshell**
change a user's login shell

This procedure allows the user to change the command run when a user logs in. The login shell of the administrative and system logins cannot be changed by this procedure.

EXAMPLES

sysadm adduser

FILES

The files that support *sysadm* are found in **/usr/admin**.

The menu starts in directory **/usr/admin/menu**.



NAME

sysdef – output system definition

SYNOPSIS

/etc/sysdef [system_namelist [master.d]]

DESCRIPTION

sysdef outputs the current system definition in tabular form. It lists all hardware devices, their local bus addresses, and unit count, as well as pseudo devices, system devices, loadable modules and the values of all tunable parameters. It generates the output by analyzing the named operating system file (*system_namelist*) and extracting the configuration information from the name list itself. The operating system file must be an "absolute" boot file (see *mkunix*[1M]).

FILES

/unix default operating system file (where the system namelist is)
/etc/master.d/* default directory containing master files

SEE ALSO

mkunix(1M), *master*(4).
nlist(3C) in the *Programmer's Reference Manual*.

DIAGNOSTICS

internal name list overflow
if the master table contains more than an internally specified number of entries for use by *nlist*(3C).



NAME

tic – terminfo compiler

SYNOPSIS

tic [-v[n]] [-c] file

DESCRIPTION

tic translates a *terminfo*(4) file from the source format into the compiled format. The results are placed in the directory */usr/lib/terminfo*. The compiled format is necessary for use with the library routines described in *curses*(3X).

- vn (verbose) output to standard error trace information showing *tic*'s progress. The optional integer *n* is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.
- c only check *file* for errors. Errors in *use=* links are not detected.
- file contains one or more *terminfo*(4) terminal descriptions in source format (see *terminfo*(4)). Each description in the file describes the capabilities of a particular terminal. When a *use=entry-name* field is discovered in a terminal entry currently being compiled, *tic* reads in the binary from */usr/lib/terminfo* to complete the entry. (Entries created from *file* will be used first. If the environment variable *TERMINFO* is set, that directory is searched instead of */usr/lib/terminfo*.) *tic* duplicates the capabilities in *entry-name* for the current entry, with the exception of those capabilities that explicitly are defined in the current entry.

If the environment variable *TERMINFO* is set, the compiled results are placed there instead of */usr/lib/terminfo*.

FILES

*/usr/lib/terminfo/?/** compiled terminal description data base

SEE ALSO

curses(3X) in the *Programmer's Reference Manual*.
term(4), *terminfo*(4) in the *System Administrator's Reference Manual*.
 Chapter 10 in the *Programmer's Guide*.

WARNINGS

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

When the *-c* option is used, duplicate terminal names will not be diagnosed; however, when *-c* is not used, they will be.

BUGS

To allow existing executables from the previous release of the UNIX System to continue to run with the compiled *terminfo* entries created by the new *terminfo* compiler, cancelled capabilities will not be marked as cancelled within the *terminfo* binary unless the entry name has a '+' within it. (Such terminal

names are only used for inclusion within other entries via a `use=` entry. Such names would not be used for real terminal names.)

For example:

```
4415+nl, kf1@, kf2@, ....
```

```
4415+base, kf1=\EOc, kf2=\EOd, ....
```

```
4415-nl|4415 terminal without keys,  
use=4415+nl, use=4415+base,
```

The above example works as expected; the definitions for the keys do not show up in the `4415-nl` entry. However, if the entry `4415+nl` did not have a plus sign within its name, the cancellations would not be marked within the compiled file and the definitions for the function keys would not be cancelled within `4415-nl`.

DIAGNOSTICS

Most diagnostic messages produced by *tic* during the compilation of the source file are preceded with the approximate line number and the name of the terminal currently being worked on.

mkdir ... returned bad status

The named directory could not be created.

File does not start with terminal names in column one

The first thing seen in the file, after comments, must be the list of terminal names.

Token after a *seek*(2) not NAMES

Somehow the file being compiled changed during the compilation.

Not enough memory for *use_list* element

or

Out of memory

Not enough free memory was available (*malloc*(3) failed).

Can't open ...

The named file could not be created.

Error in writing ...

The named file could not be written to.

Can't link ... to ...

A link failed.

Error in re-reading compiled file ...

The compiled file could not be read back in.

Premature EOF

The current entry ended prematurely.

Backspaced off beginning of line

This error indicates something wrong happened within *tic*.

Unknown Capability - "..."

The named invalid capability was found within the file.

Wrong type used for capability "..."

For example, a string capability was given a numeric value.

Unknown token type

Tokens must be followed by '@' to cancel, ',' for booleans, '#' for numbers, or '=' for strings.

"...": bad term name

or

Line ...: Illegal terminal name - "..."

Terminal names must start with a letter or digit

The given name was invalid. Names must not contain white space or slashes, and must begin with a letter or digit.

"...": terminal name too long.

An extremely long terminal name was found.

"...": terminal name too short.

A one-letter name was found.

"..." filename too long, truncating to "..."

The given name was truncated to 14 characters due to UNIX file name length limitations.

"..." defined in more than one entry. Entry being used is "...".

An entry was found more than once.

Terminal name "..." synonym for itself

A name was listed twice in the list of synonyms.

At least one synonym should begin with a letter.

At least one of the names of the terminal should begin with a letter.

Illegal character - "..."

The given invalid character was found in the input file.

Newline in middle of terminal name

The trailing comma was probably left off of the list of names.

Missing comma

A comma was missing.

Missing numeric value

The number was missing after a numeric capability.

NULL string value

The proper way to say that a string capability does not exist is to cancel it.

Very long string found. Missing comma?

self-explanatory

Unknown option. Usage is:

An invalid option was entered.

Too many file names. Usage is:

self-explanatory

"..." non-existent or permission denied
The given directory could not be written into.

"..." is not a directory
self-explanatory

"...": Permission denied
access denied.

"...": Not a directory
tic wanted to use the given name as a directory, but it already exists
as a file

SYSTEM ERROR!! Fork failed!!!
A *fork(2)* failed.

Error in following up use-links. Either there is a loop in the links or they
reference non-existent terminals. The following is a list of the entries
involved:

A *terminfo(4)* entry with a *use=name* capability either referenced a
non-existent terminal called *name* or *name* somehow referred back to
the given entry.

NAME

`uadmin` — administrative control

SYNOPSIS

`/etc/uadmin cmd fcn`

DESCRIPTION

The `uadmin` command provides control for basic administrative functions. This command is tightly coupled to the System Administration procedures and is not intended for general use. It may be invoked only by the super-user.

The arguments `cmd` (command) and `fcn` (function) are converted to integers and passed to the `uadmin` system call.

SEE ALSO

`uadmin(2)` in the *Programmer's Reference Manual*.



NAME

`unadv` – unadvertise a Remote File Sharing resource

SYNOPSIS

`unadv resource`

DESCRIPTION

`unadv` unadvertises a Remote File Sharing *resource*, which is the advertised symbolic name of a local directory, by removing it from the advertised information on the domain name server. `unadv` prevents subsequent remote mounts of that resource. It does not affect continued access through existing remote or local mounts.

An administrator at a server can unadvertise only those resources that physically reside on the local machine. A domain administrator can unadvertise any resource in the domain from the primary name server by specifying *resource* name as *domain.resource*. (A domain administrator should only unadvertise another hosts resources to clean up the domain advertise table when that host goes down. Unadvertising another host's resource changes the domain advertise table, but not the host advertise table.)

This command is restricted to the super-user.

ERRORS

If *resource* is not found in the advertised information, an error message will be sent to standard error.

SEE ALSO

`adv(1M)`, `fumount(1M)`, `nsquery(1M)`.



NAME

`uuccheck` – check the uucp directories and permissions file

SYNOPSIS

`/usr/lib/uucp/uuccheck [-v] [-x debug_level]`

DESCRIPTION

`uuccheck` checks for the presence of the `uucp` system required files and directories. Within the `uucp` makefile, it is executed before the installation takes place. It also checks for some obvious errors in the Permissions file (`/usr/lib/uucp/Permissions`). When executed with the `-v` option, it gives a detailed explanation of how the uucp programs will interpret the Permissions file. The `-x` option is used for debugging. *debug-option* is a single digit in the range 1-9; the higher the value, the greater the detail.

Note that `uuccheck` can only be used by the super-user or `uucp`.

FILES

`/usr/lib/uucp/Systems`
`/usr/lib/uucp/Permissions`
`/usr/lib/uucp/Devices`
`/usr/lib/uucp/Maxuuscheds`
`/usr/lib/uucp/Maxuuxqts`
`/usr/spool/uucp/*`
`/usr/spool/locks/LCK*`
`/usr/spool/uucppublic/*`

SEE ALSO

`uucico(1M)`, `uusched(1M)`,
`uucp(1C)`, `uustat(1C)`, `uux(1C)` in the *User's Reference Manual*.

BUGS

The program does not check file/directory modes or some errors in the Permissions file such as duplicate login or machine name.



NAME

`uucico` – file transport program for the `uucp` system

SYNOPSIS

```
/usr/lib/uucp/uucico [ -r role_number ] [ -x debug_level ]
[ -i interface ] [ -d spool_directory ] -s system_name
```

DESCRIPTION

`uucico` is the file transport program for `uucp` work file transfers. Role numbers for the `-r` are the digit 1 for master mode or 0 for slave mode (default). The `-r` option should be specified as the digit 1 for master mode when `uucico` is started by a program or `cron`. `Uux` and `uucp` both queue jobs that will be transferred by `uucico`. It is normally started by the scheduler, `uusched`, but can be started manually; this is done for debugging. For example, the shell `Uutry` starts `uucico` with debugging turned on. A single digit must be used for the `-x` option with higher numbers for more debugging.

The `-i` option defines the *interface* used with `uucico`. This interface only affects slave mode. Known interfaces are UNIX (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read/write).

The `-d` option is used to specify the directory (*spool_directory*) that contains the work files to be transferred. The default spool directory is `/usr/spool/uucp`. The `-s` option defines the system (*system_name*) that `uucico` will try to contact. The *system_name* must be defined in the **Systems** file.

FILES

```
/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Devconfig
/usr/lib/uucp/Sysfiles
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*
```

SEE ALSO

`cron(1M)`, `uusched(1M)`, `uutry(1M)`.
`uucp(1C)`, `uustat(1C)`, `uux(1C)` in the *User's Reference Manual*.



NAME

uucleanup – uucp spool directory clean-up

SYNOPSIS

```
/usr/lib/uucp/uucleanup [ -Ctime ] [ -Wtime ] [ -Xtime ] [ -mstring ]
[ -otime ] [ -ssystem ]
```

DESCRIPTION

uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way:

Inform the requestor of send/receive requests for systems that can not be reached.

Return mail, which cannot be delivered, to the sender.

Delete or execute rnews for rnews type files (depending on where the news originated—locally or remotely).

Remove all other files.

In addition, there is provision to warn users of requests that have been waiting for a given number of days (default 1). Note that *uucleanup* will process as if all option *times* were specified to the default values unless *time* is specifically set.

The following options are available.

- C*time* Any C. files greater or equal to *time* days old will be removed with appropriate information to the requestor. (default 7 days)
- D*time* Any D. files greater or equal to *time* days old will be removed. An attempt will be made to deliver mail messages and execute rnews when appropriate. (default 7 days)
- W*time* Any C. files equal to *time* days old will cause a mail message to be sent to the requestor warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (-m option). (default 1 day)
- X*time* Any X. files greater or equal to *time* days old will be removed. The D. files are probably not present (if they were, the X. could get executed). But if there are D. files, they will be taken care of by D. processing. (default 2 days)
- mstring This line will be included in the warning message generated by the -W option.
- o*time* Other files whose age is more than *time* days will be deleted. (default 2 days) The default line is "See your local administrator to locate the problem".
- ssystem Execute for *system* spool directory only.

-xdebug_level

The *-x* debug level is a single digit between 0 and 9; higher numbers give more detailed debugging information. (If **uucleanup** was compiled with *-DSMALL*, no debugging output will be available.)

This program is typically started by the shell *uudemon.cleanup*, which should be started by *cron(1M)*.

FILES

/usr/lib/uucp directory with commands used by *uucleanup* internally
/usr/spool/uucp spool directory

SEE ALSO

cron(1M).
uucp(1C), *uux(1C)* in the *User's Reference Manual*.

NAME

`uugetty` – set terminal type, modes, speed, and line discipline

SYNOPSIS

```
/usr/lib/uucp/uugetty [-t timeout] [-r] line [speed [type [linedisc] ] ]
/usr/lib/uucp/uugetty -c file
```

DESCRIPTION

`uugetty` is identical to `getty(1M)` but changes have been made to support using the line for `uucico`, `cu`, and `ct`; that is, the line can be used in both directions. The `uugetty` will allow users to login, but if the line is free, `uucico`, `cu`, or `ct` can use it for dialing out. The implementation depends on the fact that `uucico`, `cu`, and `ct` create lock files when devices are used. When the "open()" returns (or the first character is read when `-r` option is used), the status of the lock file indicates whether the line is being used by `uucico`, `cu`, `ct`, or someone trying to login. Note that in the `-r` case, several <carriage-return> characters may be required before the login message is output. The human users will be able to handle this slight inconvenience. `Uucico` trying to login will have to be told by using the following login script:

```
"" \r\d\r\d\r\d\r in:--in: . . .
```

where the . . . is whatever would normally be used for the login sequence.

If there is a `uugetty` on one end of a direct line, there must be a `uugetty` on the other end as well. Here is an `/etc/inittab` entry using `uugetty` on an intelligent modem or direct line:

```
30:2:respawn:/usr/lib/uucp/uugetty -r -t 60 tty12 1200
```

The meanings of the available options are

-t *timeout*

Specifies that `uugetty` should exit if the open on the line succeeds and there is no response to the login prompt in *timeout* seconds. *timeout* is replaced by an integer.

-r Causes `uugetty` to wait to read a character before it puts out the login message, thus preventing two `uugettys` from looping. An entry for an intelligent modem or direct line that has a `uugetty` on each end must use this option.

line Defines the name of the line to which `uugetty` will attach itself. The line name will point to an entry in the `/dev` directory. For example, `/dev/tty03`.

speed Defines the entry to use from the `/etc/gettydefs` file. The entry defines the line speed, the login message, the initial tty setting, and the next speed to try if the user says the speed is inappropriate (by sending a *break* character). The default *speed* is 300.

type Defines the type of terminal connected to the line. The default terminal is **none**, representing a normal terminal unknown to the system.

linedisc Sets the line discipline to use on the line. The default is **LDISC0**, which is the only one currently compiled into the operating system.

-c file Checks the speed and tty definitions in *file* and sends the results to standard output. Unrecognized modes and improperly constructed entries are reported. For correct entries, flag values are printed. *file* is replaced by */etc/gettydefs* or a similarly structured file.

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

uucico(1M), *getty(1M)*, *init(1M)*, *gettydefs(4)*, *inittab(4)*, *tty(7)*.
ct(1C), *cu(1C)*, *login(1)* in the *User's Reference Manual*.
ioctl(2), in the *Programmer's Reference Manual*.

BUGS

ct will not work when *uugetty* is used with an intelligent modem such as *penril* or *ventel*.

NAME

`uusched` – the scheduler for the `uucp` file transport program

SYNOPSIS

```
/usr/lib/uucp/uusched [ -x debug_level ] [ -u debug_level ]
```

DESCRIPTION

`uusched` is the `uucp` file transport scheduler. It is usually started by the daemon `uudemon.hour` that is started by `cron(1M)` from an entry in `/usr/spool/cron/crontab`:

```
39 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour > /dev/null"
```

The two options are for debugging purposes only; `-x debug_level` will output debugging messages from `uusched` and `-u debug_level` will be passed as `-x debug_level` to `uucico`. The `debug_level` is a number between 0 and 9; higher numbers give more detailed information.

FILES

```
/usr/lib/uucp/Systems  
/usr/lib/uucp/Permissions  
/usr/lib/uucp/Devices  
/usr/spool/uucp/*  
/usr/spool/locks/LCK*  
/usr/spool/uucppublic/*
```

SEE ALSO

`cron(1M)`, `uucico(1M)`,
`uucp(1C)`, `uustat(1C)`, `uux(1C)` in the *User's Reference Manual*.



NAME

Uutry – try to contact remote system with debugging on

SYNOPSIS

`/usr/lib/uucp/Uutry [-x debug_level] [-r] system_name`

DESCRIPTION

Uutry is a shell that is used to invoke *uucico* to call a remote site. Debugging is turned on (default is level 5); `-x` will override that value. The `-r` overrides the retry time in `/usr/spool/uucp/.status`. The debugging output is put in file `/tmp/system_name`. A tail `-f` of the output is executed. A `<DELETE>` or `<BREAK>` will give control back to the terminal while the *uucico* continues to run, putting its output in `/tmp/system_name`.

FILES

`/usr/lib/uucp/Systems`
`/usr/lib/uucp/Permissions`
`/usr/lib/uucp/Devices`
`/usr/lib/uucp/Maxuuxqts`
`/usr/lib/uucp/Maxuuscheds`
`/usr/spool/uucp/*`
`/usr/spool/locks/LCK*`
`/usr/spool/uucppublic/*`
`/tmp/system_name`

SEE ALSO

`uucico(1M)`.
`uucp(1C)`, `uux(1C)` in the *User's Reference Manual*.



NAME

`uuxqt` – execute remote command requests

SYNOPSIS

```
/usr/lib/uucp/uuxqt [ -s system ] [ -x debug_level ]
```

DESCRIPTION

`uuxqt` is the program that executes remote job requests from remote systems generated by the use of the `uux` command. (*Mail* uses `uux` for remote mail requests). `uuxqt` searches the spool directories looking for *X* files. For each *X* file, `uuxqt` checks to see if all the required data files are available and accessible, and file commands are permitted for the requesting system. The *Permissions* file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the `uuxqt` command is executed:

`UU_MACHINE` is the machine that sent the job (the previous one).

`UU_USER` is the user that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

The `-x debug_level` is a single digit between 0 and 9. Higher numbers give more detailed debugging information.

FILES

```
/usr/lib/uucp/Permissions  
/usr/lib/uucp/Maxuuxqts  
/usr/spool/uucp/*  
/usr/spool/locks/LCK*
```

SEE ALSO

`uucico(1M)`.

`uucp(1C)`, `uustat(1C)`, `uux(1C)`, `mail(1)` in the *User's Reference Manual*.



NAME

volcopy – make literal copy of file system

SYNOPSIS

/etc/volcopy [options] *fsname srcdevice volname1 destdevice volname2*

DESCRIPTION

volcopy makes a literal copy of the file system using a blocksize matched to the device. *Options* are:

- a invoke a verification sequence requiring a positive operator response instead of the standard 10 second delay before the copy is made
- s (default) invoke the DEL if wrong verification sequence.

The program requests length and density information if it is not given on the command line or is not recorded on an input tape label. If the file system is too large to fit on one reel, *volcopy* will prompt for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two or more drives. If *volcopy* is interrupted, it will ask if the user wants to quit or wants a shell. In the latter case, the user can perform other operations (e.g., *labelit*) and return to *volcopy* by exiting the new shell.

The *fsname* argument represents the mounted name (e.g.: *root*, *u1*, etc.) of the filesystem being copied.

The *srcdevice* or *destdevice* should be the physical disk section or tape (e.g.: */dev/dsk/c1d0s8*, */dev/rdisk/c1d1s8*, etc.).

The *volname* is the physical volume name (e.g.: *pk3*, *t0122*, etc.) and should match the external label sticker. Such label names are limited to six or fewer characters. *Volname* may be – to use the existing volume name.

Srcdevice and *volname1* are the device and volume from which the copy of the file system is being extracted. *Destdevice* and *volname2* are the target device and volume.

Fsname and *volname* are recorded in the last 12 characters of the superblock (*char fsname[6]*, *volname[6]*).

FILES

/etc/log/filesave.log a record of file systems/volumes copied

SEE ALSO

labelit(1M), *fs(4)*,
sh(1) in the *User's Reference Manual*.

WARNINGS

volcopy does not support tape-to-tape copying. Use *dd(1)* for tape-to-tape copying.



NAME

whodo – who is doing what

SYNOPSIS

/etc/whodo

DESCRIPTION

whodo produces formatted and dated output from information in the */etc/utmp* and */etc/ps_data* files.

The display is headed by the date, time and machine name. For each user logged in, device name, user-id and login time is shown, followed by a list of active processes associated with the user-id. The list includes the device name, process-id, cpu minutes and seconds used, and process name.

EXAMPLE

The command:

```
whodo
```

produces a display like this:

```
Tue Mar 12 15:48:03 1985
bailey

tty09  mcn      8:51
      tty09  28158  0:29 sh

tty52  bdr      15:23
      tty52  21688  0:05 sh
      tty52  22788  0:01 whodo
      tty52  22017  0:03 vi
      tty52  22549  0:01 sh

xt162  lee      10:20
      tty08  6748   0:01 layers
      xt162  6751   0:01 sh
      xt163  6761   0:05 sh
      tty08  6536   0:05 sh
```

FILES

/etc/passwd
/etc/ps_data
/etc/utmp

SEE ALSO

ps(1), *who(1)* in the *User's Reference Manual*.



NAME

`wtinit` – object downloader for the 5620 DMD terminal

SYNOPSIS

`/usr/lib/layersys/wtinit [-d] [-p] file`

DESCRIPTION

The `wtinit` utility downloads the named *file* for execution in the AT&T Teletype 5620 DMD terminal connected to its standard output. *file* must be a DMD object file. `wtinit` performs all necessary bootstrap and protocol procedures.

There are two options.

- `-d` Prints out the sizes of the text, data, and bss portions of the downloaded *file* on standard error.
- `-p` Prints the down-loading protocol statistics and a trace on standard error.

The environment variable `JPATH` is the analog of the shell's `PATH` variable to define a set of directories in which to search for *file*.

If the environment variable `DMDLOAD` has the value `hex`, `wtinit` will use a hexadecimal download protocol that uses only printable characters.

Terminal Feature Packages for specific versions of AT&T windowing terminals will include terminal-specific versions of `wtinit` under those installation sub-directories. `/usr/lib/layersys/wtinit` is used for `layers(1)` initialization only when no Terminal Feature Package is in use.

EXIT STATUS

Returns 0 upon successful completion, 1 otherwise.

WARNING

Standard error should be redirected when using the `-d` or `-p` options.

SEE ALSO

`layers(1)` in the *User's Reference Manual*.



NAME

`xtd` – extract and print `xt` driver link structure

SYNOPSIS

`xtd` [-f] [-n ...]

DESCRIPTION

The `xtd` command is a debugging tool for the `xt(7)` driver. It performs an `XTIOCDATA ioctl(2)` call on its standard input file to extract the *Link* data structure for the attached group of channels. This call will fail if data extraction has not been configured in the driver or the standard input is not attached to an `xt(7)` channel. The data are printed one item per line on the standard output. The output should probably be formatted via `pr -3`.

The optional flags affect output as follows:

- n *n* is a number in the range 0 to 7. Channel *n* is included in the list of channels to be printed. The default prints all channels, whereas the occurrence of one or more channel numbers implies a subset.
- f Causes a “formfeed” character to be put out at the end of the output, for the benefit of page-display programs.

DIAGNOSTICS

Returns 0 upon successful completion, 1 otherwise.

SEE ALSO

`xts(1M)`, `xtt(1M)`, `ioctl(2)` in the *Programmer's Reference Manual*.
`xtproto(5)`, `xt(7)` in the *System Administrator's Reference Manual*.
`pr(1)` in the *User's Reference Manual*.



NAME

`xts` – extract and print `xt` driver statistics

SYNOPSIS

`xts` [-f]

DESCRIPTION

The `xts` command is a debugging tool for the `xt(7)` driver. It performs an `XTIOCTSTATS ioctl(2)` call on its standard input file to extract the accumulated statistics for the attached group of channels. This call will fail if statistics have not been configured in the driver or the standard input is not attached to an `xt(7)` channel. The statistics are printed one item per line on the standard output.

-f Causes a “formfeed” character to be put out at the end of the output, for the benefit of page-display programs.

DIAGNOSTICS

Returns 0 upon successful completion, 1 otherwise.

SEE ALSO

`xtd(1M)`, `xtd(1M)`, `ioctl(2)` in the *Programmer's Reference Manual*.
`xtproto(5)`, `xt(7)` in the *System Administrator's Reference Manual*.



NAME

`xtt` – extract and print `xt` driver packet traces

SYNOPSIS

`xtt` [-f] [-o]

DESCRIPTION

The `xtt` command is a debugging tool for the `xt(7)` driver. It performs an `XTIOCTRACE ioctl(2)` call on its standard input file to turn on tracing and extract the circular packet trace buffer for the attached group of channels. This call will fail if tracing has not been configured in the driver, or the standard input is not attached to an `xt(7)` channel. The packets are printed on the standard output.

The optional flags are:

- f Causes a “formfeed” character to be put out at the end of the output, for the benefit of page-display programs.
- o Turns off further driver tracing.

DIAGNOSTICS

Returns 0 upon successful completion, 1 otherwise.

NOTE

If driver tracing has not been turned on for the terminal session by invoking `layers(1)` with the `-t` option, `xtt` will not generate any output the first time it is executed.

SEE ALSO

`xtd(1M)`, `xts(1M)`, `ioctl(2)` in the *Programmer's Reference Manual*.
`layers(5)`, `xt(7)` in the *System Administrator's Reference Manual*.
`layers(1)` in the *User's Reference Manual*.



NAME

intro – introduction to file formats

DESCRIPTION

This section outlines the formats of various files. The C structure declarations for the file formats are given where applicable. Usually, the header files containing these structure declarations can be found in the directories `/usr/include` or `/usr/include/sys`. For inclusion in C language programs, however, the syntax `#include <filename.h>` or `#include <sys/filename.h>` should be used.



NAME

a.out – common assembler and link editor output

SYNOPSIS

```
#include <a.out.h>
```

DESCRIPTION

The file name **a.out** is the default output file name from the link editor *ld(1)*. The link editor will make *a.out* executable if there were no errors in linking. The output file of the assembler *as(1)*, also follows the common object file format of the *a.out* file although the default file name is different.

A common object file consists of a file header, a UNIX system header (if the file is link editor output), a table of section headers, relocation information, (optional) line numbers, a symbol table, and a string table. The order is given below.

```
File header.
UNIX system header.
Section 1 header.
...
Section n header.
Section 1 data.
...
Section n data.
Section 1 relocation.
...
Section n relocation.
Section 1 line numbers.
...
Section n line numbers.
Symbol table.
String table.
```

The last three parts of an object file (line numbers, symbol table and string table) may be missing if the program was linked with the `-s` option of *ld(1)* or if they were removed by *strip(1)*. Also note that the relocation information will be absent after linking unless the `-r` option of *ld(1)* was used. The string table exists only if the symbol table contains symbols with names longer than eight characters.

The sizes of each section (contained in the header, discussed below) are in bytes.

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. On the 3B2 computer the text segment starts at location 0x80800000.

The **a.out** file produced by *ld(1)* has the magic number 0413 in the first field of the UNIX system header. The headers (file header, UNIX system header, and section headers) are loaded at the beginning of the text segment and the text

immediately follows the headers in the user address space. The first text address will equal 0x80800000 plus the size of the headers, and will vary depending upon the number of section headers in the **a.out** file. In an **a.out** file with three sections (**.text**, **.data**, and **.bss**), the first text address is at 0x808000A8 on the 3B2 computer. 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 512K boundary past the last text address. The first data address is determined by the following: If an **a.out** file were split into 8K chunks, one of the chunks would contain both the end of text and the beginning of data. When the core image is created, that chunk will appear twice; once at the end of text and once at the beginning of data (with some unused space in between). 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 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 segment boundary past the end of text plus the remainder of the last text address divided by 8K. If the last text address is a multiple of 8K no duplication is necessary.

On the 3B2 computer the stack begins at location 0xC0020000 and grows toward higher addresses. The stack is automatically extended as required. The data segment is extended only as requested by the **brk(2)** system call.

For relocatable files the value of a word in the text or data portions that is not a reference to an undefined external symbol is exactly the value that will appear in memory when the file is executed. If a word in the text involves a reference to an undefined external symbol, there will be a relocation entry for the word, the storage class of the symbol-table entry for the symbol will be marked as an "external symbol", and the value and section number of the symbol-table entry will be undefined. When the file is processed by the link editor and the external symbol becomes defined, the value of the symbol will be added to the word in the file.

File Header

The format of the **filehdr** header is

```

struct filehdr
{
    unsigned short f_magic; /* magic number */
    unsigned short f_nscns; /* number of sections */
    long          f_timdat; /* time and date stamp */
    long          f_symptr; /* file ptr to symtab */
    long          f_nsyms; /* # symtab entries */
    unsigned short f_opthdr; /* sizeof(opt hdr) */
    unsigned short f_flags; /* flags */
};

```

UNIX System Header

The format of the UNIX system header is

```
typedef struct aouthdr
{
    short    magic;           /* magic number */
    short    vstamp;         /* version stamp */
    long     tsize;          /* text size in bytes, padded */
    long     dsize;          /* initialized data (.data) */
    long     bsize;          /* uninitialized data (.bss) */
    long     entry;          /* entry point */
    long     text_start;     /* base of text used for this file */
    long     data_start;     /* base of data used for this file */
} AOUTHDR;
```

Section Header

The format of the section header is

```
struct schdr
{
    char     s_name[SYMNMLEN]; /* section name */
    long     s_paddr;          /* physical address */
    long     s_vaddr;          /* virtual address */
    long     s_size;           /* section size */
    long     s_scnptr;         /* file ptr to raw data */
    long     s_relptr;         /* file ptr to relocation */
    long     s_lnnoptr;        /* file ptr to line numbers */
    unsigned short s_nreloc;   /* # reloc entries */
    unsigned short s_nlnno;    /* # line number entries */
    long     s_flags;          /* flags */
};
```

Relocation

Object files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format:

```
struct reloc
{
    long     r_vaddr;         /* (virtual) address of reference */
    long     r_symndx;        /* index into symbol table */
    ushort   r_type;          /* relocation type */
};
```

The start of the relocation information is *s_relptr* from the section header. If there is no relocation information, *s_relptr* is 0.

Symbol Table

The format of each symbol in the symbol table is

```

#define SYMNMLEN 8
#define FILNMLEN 14
#define DIMNUM 4

struct syment
{
    union /* all ways to get a symbol name */
    {
        char _n_name[SYMNMLEN]; /* name of symbol */
        struct
        {
            long _n_zeroes; /* == 0L if in string table */
            long _n_offset; /* location in string table */
        } _n_n;
        char *_n_nptr[2]; /* allows overlaying */
    } _n;
    long n_value; /* value of symbol */
    short n_scnm; /* section number */
    unsigned short n_type; /* type and derived type */
    char n_sclass; /* storage class */
    char n_numaux; /* number of aux entries */
};

#define n_name _n._n_name
#define n_zeroes _n._n_n._n_zeroes
#define n_offset _n._n_n._n_offset
#define n_nptr _n._n_nptr[1]

```

Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows.

```

union auxent {
    struct {
        long    x_tagndx;
        union {
            struct {
                unsigned short x_inno;
                unsigned short x_size;
            } x_insz;
            long    x_fsize;
        } x_misc;
        union {
            struct {
                long    x_innoptr;
                long    x_endndx;
            } x_fcn;
            struct {
                unsigned short x_dimen[DIMNUM];
            } x_ary;
        } x_fcary;
        unsigned short x_tvndx;
    } x_sym;

    struct {
        char    x_fname[FILNMLEN];
    } x_file;

    struct {
        long    x_scnlen;
        unsigned short x_nreloc;
        unsigned short x_nlinno;
    } x_scn;

    struct {
        long    x_tvfill;
        unsigned short x_tvlen;
        unsigned short x_tvran[2];
    } x_tv;
};

```

Indexes of symbol table entries begin at *zero*. The start of the symbol table is *f_symptr* (from the file header) bytes from the beginning of the file. If the symbol table is stripped, *f_symptr* is 0. The string table (if one exists) begins at *f_symptr + (f_nsyms * SYMESZ)* bytes from the beginning of the file.

SEE ALSO

as(1), cc(1), ld(1), brk(2) in the *Programmer's Reference Manual*.
filehdr(4), ldfcn(4), linenum(4), reloc(4), scnhdr(4), syms(4).



NAME

acct – per-process accounting file format

SYNOPSIS

```
#include <sys/acct.h>
```

DESCRIPTION

Files produced as a result of calling *acct(2)* have records in the form defined by *<sys/acct.h>*, whose contents are:

```
typedef ushort comp_t; /* "floating point" */
                        /* 13-bit fraction, 3-bit exponent */

struct acct
{
    char  ac_flag;      /* Accounting flag */
    char  ac_stat;     /* Exit status */
    ushort ac_uid;     /* Accounting user ID */
    ushort ac_gid;     /* Accounting group ID */
    dev_t ac_tty;      /* control typewriter */
    time_t ac_btime;   /* Beginning time */
    comp_t ac_utime;   /* acctng user time in clock ticks */
    comp_t ac_stime;   /* acctng system time in clock ticks */
    comp_t ac_etime;   /* acctng elapsed time in clock ticks */
    comp_t ac_mem;     /* memory usage in clicks */
    comp_t ac_io;      /* chars trnsfrd by read/write */
    comp_t ac_rw;      /* number of block reads/writes */
    char  ac_comm[8];  /* command name */
}

extern struct acct      acctbuf;
extern struct inode    *acctp; /* inode of accounting file */

#define AFORK 01      /* has executed fork, but no exec */
#define ASU  02      /* used super-user privileges */
#define ACCTF 0300    /* record type: 00 = acct */
```

In *ac_flag*, the *AFORK* flag is turned on by each *fork(2)* and turned off by an *exec(2)*. The *ac_comm* field is inherited from the parent process and is reset by any *exec*. Each time the system charges the process with a clock tick, it also adds to *ac_mem* the current process size, computed as follows:

$$(\text{data size}) + (\text{text size}) / (\text{number of in-core processes using text})$$

The value of $ac_mem / (ac_stime + ac_utime)$ can be viewed as an approximation to the mean process size, as modified by text sharing.

The structure `acct`, which resides with the source files of the accounting commands, represents the total accounting format used by the various accounting commands:

```
/*
 * total accounting (for acct period), also for day
 */

struct tacct {
    uid_t      ta_uid;      /* userid */
    char       ta_name[8]; /* login name */
    float      ta_cpu[2];  /* cum. cpu time, p/np (mins) */
    float      ta_kcore[2]; /* cum kcore-minutes, p/np */
    float      ta_con[2];  /* cum. connect time, p/np, mins */
    float      ta_du;      /* cum. disk usage */
    long       ta_pc;      /* count of processes */
    unsigned short ta_sc;  /* count of login sessions */
    unsigned short ta_dc;  /* count of disk samples */
    unsigned short ta_fee; /* fee for special services */
};
```

SEE ALSO

`acct(2)`, `exec(2)`, `fork(2)` in the *Programmer's Reference Manual*.
`acct(1M)` in the *System Administrator's Reference Manual*.
`acctcom(1)` in the *User's Reference Manual*.

BUGS

The `ac_mem` value for a short-lived command gives little information about the actual size of the command, because `ac_mem` may be incremented while a different command (e.g., the shell) is being executed by the process.

NAME

pathalias – alias file for FACE

DESCRIPTION

The pathalias files contain lines of the form "alias=path" where "path" can be one or more colon (:) separated directories. Whenever a FACE 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 UNIX 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.

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.

FILES

\$HOME/pref/pathalias
\$VMSYS/pathalias



NAME

ar – common archive file format

SYNOPSIS

```
#include <ar.h>
```

DESCRIPTION

The archive command *ar(1)* is used to combine several files into one. Archives are used mainly as libraries to be searched by the link editor *ld(1)*.

Each archive begins with the archive magic string.

```
#define ARMAG "!<arch>\n"      /* magic string */
#define SARMAG 8                /* length of magic string */
```

Each archive which contains common object files [see *a.out(4)*] includes an archive symbol table. This symbol table is used by the link editor *ld(1)* to determine which archive members must be loaded during the link edit process. The archive symbol table (if it exists) is always the first file in the archive (but is never listed) and is automatically created and/or updated by *ar*.

Following the archive magic string are the archive file members. Each file member is preceded by a file member header which is of the following format:

```
#define ARFMAG  "\n"          /* header trailer string */

struct ar_hdr          /* file member header */
{
    char    ar_name[16];    /* '/' terminated file member name */
    char    ar_date[12];    /* file member date */
    char    ar_uid[6];      /* file member user identification */
    char    ar_gid[6];      /* file member group identification */
    char    ar_mode[8];     /* file member mode (octal) */
    char    ar_size[10];    /* file member size */
    char    ar_fmag[2];     /* header trailer string */
};
```

All information in the file member headers is in printable ASCII. The numeric information contained in the headers is stored as decimal numbers (except for *ar_mode* which is in octal). Thus, if the archive contains printable files, the archive itself is printable.

The *ar_name* field is blank-padded and slash (/) terminated. The *ar_date* field is the modification date of the file at the time of its insertion into the archive. Common format archives can be moved from system to system as long as the portable archive command *ar(1)* is used. Conversion tools such as *convert(1)* exist to aid in the transportation of non-common format archives to this format.

Each archive file member begins on an even byte boundary; a newline is inserted between files if necessary. Nevertheless the size given reflects the actual size of the file exclusive of padding.

Notice there is no provision for empty areas in an archive file.

If the archive symbol table exists, the first file in the archive has a zero length name (i.e., `ar_name[0] == '/'`). The contents of this file are as follows:

- The number of symbols. Length: 4 bytes.
- The array of offsets into the archive file. Length: 4 bytes * "the number of symbols".
- The name string table. Length: $ar_size - (4 \text{ bytes} * (\text{"the number of symbols"} + 1))$.

The number of symbols and the array of offsets are managed with `sgetl` and `sputl`. The string table contains exactly as many null terminated strings as there are elements in the offsets array. Each offset from the array is associated with the corresponding name from the string table (in order). The names in the string table are all the defined global symbols found in the common object files in the archive. Each offset is the location of the archive header for the associated symbol.

SEE ALSO

`ar(1)`, `ld(1)`, `strip(1)`, `sputl(3X)` in the *Programmer's Reference Manual*.
`a.out(4)`.

WARNINGS

`Strip(1)` will remove all archive symbol entries from the header. The archive symbol entries must be restored via the `ts` option of the `ar(1)` command before the archive can be used with the link editor `ld(1)`.

NAME

cftime – language specific strings

DESCRIPTION

The programmer can create one printable file per language. These files must be kept in a special directory `/lib/cftime`. If this directory does not exist, the programmer should create it. The contents of these files are:

- abbreviated month names (in order)
- month names (in order)
- abbreviated weekday names (in order)
- weekday names (in order)
- default strings that specify formats for local time (%x) and local date (%X).
- default format for cftime, if the argument for cftime is zero or null.
- AM (ante meridian) string
- PM (post meridian) string

Each string is on a line by itself. All white space is significant. The order of the strings in the above list is the same order in which the strings appear in the file shown below.

EXAMPLE

```

/lib/cftime/usa_english
Jan
Feb
...
January
February
...
Sun
Mon
...
Sunday
Monday
...
%H:%M:%S
%m/%d/%y
%a %b %d %T %Z %Y
AM
PM

```

FILES

`/lib/cftime` – directory that contains the language specific printable files (create it if it does not exist)

SEE ALSO

cftime(3C) in the *Programmer's Reference Manual*.



NAME

checklist – list of file systems processed by fsck and ncheck

DESCRIPTION

checklist resides in directory */etc* and contains a list of, at most, 15 *special file* names. Each *special file* name is contained on a separate line and corresponds to a file system. Each file system will then be automatically processed by the *fsck(1M)* command.

FILES

/etc/checklist

SEE ALSO

fsck(1M), *ncheck(1M)* in the *System Administrator's Reference Manual*.



NAME

core – format of core image file

DESCRIPTION

The UNIX system writes out a core image of a terminated process when any of various errors occur. See *signal(2)* for the list of reasons; the most common are memory violations, illegal instructions, bus errors, and user-generated quit 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.

The first section of the core image is a copy of the system's per-user data for the process, including the registers as they were at the time of the fault. The size of this section depends on the parameter *usize*, which is defined in `<sys/param.h>`. The remainder represents the actual contents of the user's core area when the core image was written. If the text segment is read-only and shared, or separated from data space, it is not dumped.

The format of the information in the first section is described by the *user* structure of the system, defined in `<sys/user.h>`. Not included in this file are the locations of the registers. These are outlined in `<sys/reg.h>`.

SEE ALSO

crash(1M).

sdb(1), setuid(2), signal(2) in the *Programmer's Reference Manual*.



NAME

cpio – format of cpio archive

DESCRIPTION

The *header* structure, when the `-c` option of *cpio*(1) is not used, is:

```

struct {
    short    h_magic,
            h_dev;
    ushort   h_ino,
            h_mode,
            h_uid,
            h_gid;
    short    h_nlink,
            h_rdev,
            h_mtime[2],
            h_namesize,
            h_filesize[2];
    char     h_name[h_namesize rounded to word];
} Hdr;

```

When the `-c` option is used, the *header* information is described by:

```

sscanf(Chdr,"%6o%6o%6o%6o%6o%6o%6o%6o%11lo%6o%11lo%s",
        &Hdr.h_magic, &Hdr.h_dev, &Hdr.h_ino, &Hdr.h_mode,
        &Hdr.h_uid, &Hdr.h_gid, &Hdr.h_nlink, &Hdr.h_rdev,
        &Longtime, &Hdr.h_namesize,&Longfile,Hdr.h_name);

```

Longtime and *Longfile* are equivalent to *Hdr.h_mtime* and *Hdr.h_filesize*, respectively. The contents of each file are recorded in an element of the array of varying length structures, *archive*, together with other items describing the file. Every instance of *h_magic* contains the constant 070707 (octal). The items *h_dev* through *h_mtime* have meanings explained in *stat*(2). The length of the null-terminated path name *h_name*, including the null byte, is given by *h_namesize*.

The last record of the *archive* always contains the name TRAILER!!!. Special files, directories, and the trailer are recorded with *h_filesize* equal to zero.

SEE ALSO

stat(2) in the *Programmer's Reference Manual*.
cpio(1), *find*(1) in the *User's Reference Manual*.



NAME

dir – format of directories

SYNOPSIS

```
#include <sys/dir.h>
```

DESCRIPTION

A directory behaves exactly like an ordinary file, save that no user may write into a directory. The fact that a file is a directory is indicated by a bit in the flag word of its i-node entry [see *fs(4)*]. The structure of a directory entry as given in the include file is:

```
#ifndef DIRSIZ
#define DIRSIZ    14
#endif
struct direct
{
    ushortd_ino;
    char d_name[DIRSIZ];
};
```

By convention, the first two entries in each directory are for . and .. The first is an entry for the directory itself. The second is for the parent directory. The meaning of .. is modified for the root directory of the master file system; there is no parent, so .. has the same meaning as ..

SEE ALSO

fs(4).



NAME

dirent – file system independent directory entry

SYNOPSIS

```
#include <sys/dirent.h>
#include <sys/types.h>
```

DESCRIPTION

Different file system types may have different directory entries. The *dirent* structure defines a file system independent directory entry, which contains information common to directory entries in different file system types. A set of these structures is returned by the *getdents(2)* system call.

The *dirent* structure is defined below.

```
struct dirent {
                long           d_ino;
                off_t          d_off;
                unsigned short d_reclen;
                char            d_name[1];
};
```

The *d_ino* is a number which is unique for each file in the file system. The field *d_off* is the offset of that directory entry in the actual file system directory. The field *d_name* is the beginning of the character array giving the name of the directory entry. This name is null terminated and may have at most MAXNAMLEN characters. This results in file system independent directory entries being variable length entities. The value of *d_reclen* is the record length of this entry. This length is defined to be the number of bytes between the current entry and the next one, so that it will always result in the next entry being on a long boundary.

FILES

/usr/include/sys/dirent.h

SEE ALSO

getdents(2) in the *Programmer's Reference Manual*.



NAME

.edt_swapp – software application file

DESCRIPTION

The *.edt_swapp* file is read by *filledt(8)* on its second pass during the self-configuration process to rename specified Equipped Device Table (EDT) entries. The file has the following format:

SLOT	SWNAME	HWNAME
.	.	.
.	.	.
.	.	.
FF		

The number in the slot field specifies the entry in the EDT to be updated. The *SWNAME* column contains the new name which will be associated with this board. The *HWNAME* field contains the name which the board was associated with generically. The last line contains 'FF' for the slot number to signal the end of the data to the firmware. This file can be displayed by the *editsa -l* command.

WARNINGS

This file is not to be edited directly. Updates to it must be done through *editsa*.

SEE ALSO

editsa(1M), *filledt(8)*.



NAME

.environ – system-wide FACE environment variables
 .pref - default preferences for WASTEBASKET and FILECABINET
 .variables

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".

Variables found in .environ include:

LOGINWIN1 - Windows that are opened when FACE is initialized

...

LOGINWIN4

SORTMODE - Sort mode for file folder listings.

Values include the following hexadecimal digits:

- 1 sorted alphabetically by name
- 2 files most recently modified first
- 800 sorted alphabetically by object type

The values above may be listed in reverse order by "ORing" the following value:

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 "reverse" alphabetical by name listing of the folder

DISPLAYMODE - Display mode for file folders

Values include the following hexadecimal digits:

- 0 file names only
- 4 file names and brief description
- 8 file names, description, plus additional information

WASTEPROMPT - Prompt before emptying wastebasket (yes/no)?

WASTEDAYS - # days before emptying wastebasket

PRINCMD1 - print command defined to print files.

...

PRINCMD3

UMASK - holds default permissions that files will be created with.

Variables found in .pref are **SORTMODE** and **DISPMODE**, which have the same values as the **SORTMODE** and **DISPLAYMODE** variables described in .environ above.

Variables found in .variables include:

EDITOR - Default editor
PS1 - UNIX shell prompt

FILES

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

NAME

filehdr – file header for common object files

SYNOPSIS

```
#include <filehdr.h>
```

DESCRIPTION

Every common object file begins with a 20-byte header. The following C struct declaration is used:

```
struct filehdr
{
    unsigned short f_magic ; /* magic number */
    unsigned short f_nscns ; /* number of sections */
    long          f_timdat ; /* time & date stamp */
    long          f_symptr ; /* file ptr to symtab */
    long          f_nsyms ; /* # symtab entries */
    unsigned short f_opthdr ; /* sizeof(opt hdr) */
    unsigned short f_flags ; /* flags */
};
```

F_symptr is the byte offset into the file at which the symbol table can be found. Its value can be used as the offset in *fseek(3S)* to position an I/O stream to the symbol table. The UNIX system optional header is 28-bytes. The valid magic numbers are given below:

```
#define FBOMAGIC    0560 /* 3B2 and 3B5 computers */
#define N3BMAGIC    0550 /* 3B20 computer */
#define NTVMAGIC    0551 /* 3B20 computer */

#define VAXWRMAGIC  0570 /* VAX writable text segments */
#define VAXROMAGIC  0575 /* VAX read only sharable
                        text segments */
```

The value in *f_timdat* is obtained from the *time(2)* system call. Flag bits currently defined are:

```
#define F_RELFLG    0000001 /* relocation entries stripped */
#define F_EXEC      0000002 /* file is executable */
#define F_LNNO      0000004 /* line numbers stripped */
#define F_LSyms     0000010 /* local symbols stripped */
#define F_MINMAL    0000020 /* minimal object file */
#define F_UPDATE    0000040 /* update file, ogen produced */
#define F_SWABD     0000100 /* file is "pre-swabbed" */
#define F_AR16WR    0000200 /* 16-bit DEC host */
#define F_AR32WR    0000400 /* 32-bit DEC host */
#define F_AR32W     0001000 /* non-DEC host */
#define F_PATCH     0002000 /* "patch" list in opt hdr */
#define F_BM32ID    0160000 /* WE32000 family ID field */
#define F_BM32B     0020000 /* file contains WE 32100 code */
#define F_BM32MAU   0040000 /* file reqs MAU to execute */
```

FILEHDR(4)

FILEHDR(4)

```
#define F_BM32RST 0010000 /* this object file contains restore  
work around [3B5/3B2 only] */
```

SEE ALSO

time(2), fseek(3S) in the *Programmer's Reference Manual*.
a.out(4).

NAME

fs: file system – format of system volume

SYNOPSIS

```
#include <sys/filsys.h>
#include <sys/types.h>
#include <sys/param.h>
```

DESCRIPTION

Every file system storage volume has a common format for certain vital information. Every such volume is divided into a certain number of 512-byte long sectors. Sector 0 is unused and is available to contain a bootstrap program or other information.

Sector 1 is the *super-block*. The format of a super-block is:

```
struct   filsys
{
    ushort   s_ysize;           /* size in blocks of i-list */
    daddr_t  s_fsize;           /* size in blocks of entire volume */
    short    s_nfree;           /* number of addresses in s_free */
    daddr_t  s_free[NICFREE];   /* free block list */
    short    s_ninode;          /* number of i-nodes in s_inode */
    ushort   s_inode[NICINOD]; /* free i-node list */
    char     s_flock;           /* lock during free list manipulation */
    char     s_ilock;           /* lock during i-list manipulation */
    char     s_fmmod;           /* super block modified flag */
    char     s_ronly;           /* mounted read-only flag */
    time_t   s_time;            /* last super block update */
    short    s_dinfo[4];        /* device information */
    daddr_t  s_tfree;           /* total free blocks */
    ushort   s_tinode;          /* total free i-nodes */
    char     s_fname[6];        /* file system name */
    char     s_fpack[6];        /* file system pack name */
    long     s_fill[12];        /* ADJUST to make sizeof filsys be 512 */
    long     s_state;           /* file system state */
    long     s_magic;           /* magic number to denote new file
                                system */
    long     s_type;            /* type of new file system */
};

#define FsMAGIC  0xfd187e20    /* s_magic number */
#define Fs1b     1             /* 512-byte block */
#define Fs2b     2             /* 1024-byte block */
#define Fs4b     3             /* 2048-byte block */

#define FsOKAY   0x7c269d38    /* s_state: clean */
#define FsACTIVE 0x5e72d81a    /* s_state: active */
#define FsBAD    0xcb096f43    /* s_state: bad root */
#define FsBADBLK 0xbadbc14b    /* s_state: bad block corrupted it */
```

S_type indicates the file system type. Currently, three types of file systems are supported: the original 512-byte logical block, the 1024-byte logical block, and the 2048-byte logical block. A block size of 2048 may be used only if the 2K file system package is installed. *S_magic* is used to distinguish the original 512-byte oriented file systems from the newer file systems. If this field is not equal to the magic number, *fsMAGIC*, the type is assumed to be *fs1b*, otherwise the *s_type* field is used. In the following description, a block is then determined by the type. For the original 512-byte oriented file system, a block is 512-bytes. For the new file system types, block size depends on *s_type*. The operating system takes care of all conversions from logical block numbers to physical sector numbers.

S_state indicates the state of the file system. A cleanly unmounted, not damaged file system is indicated by the FsOKAY state. After a file system has been mounted for update, the state changes to FsACTIVE. A special case is used for the root file system. If the root file system appears damaged at boot time, it is mounted but marked FsBAD. Lastly, after a file system has been unmounted, the state reverts to FsOKAY.

S_ishize is the address of the first data block after the i-list; the i-list starts just after the super-block, namely in block 2; thus the i-list is *s_ishize*-2 blocks long. *S_fsize* is the first block not potentially available for allocation to a file. These numbers are used by the system to check for bad block numbers; if an "impossible" block number is allocated from the free list or is freed, a diagnostic is written on the on-line console. Moreover, the free array is cleared, so as to prevent further allocation from a presumably corrupted free list.

The free list for each volume is maintained as follows. The *s_free* array contains, in *s_free*[1], ..., *s_free*[*s_nfree*-1], up to 49 numbers of free blocks. *S_free*[0] is the block number of the head of a chain of blocks constituting the free list. The first long in each free-chain block is the number (up to 50) of free-block numbers listed in the next 50 longs of this chain member. The first of these 50 blocks is the link to the next member of the chain. To allocate a block: decrement *s_nfree*, and the new block is *s_free*[*s_nfree*]. If the new block number is 0, there are no blocks left, so give an error. If *s_nfree* became 0, read in the block named by the new block number, replace *s_nfree* by its first word, and copy the block numbers in the next 50 longs into the *s_free* array. To free a block, check if *s_nfree* is 50; if so, copy *s_nfree* and the *s_free* array into it, write it out, and set *s_nfree* to 0. In any event set *s_free*[*s_nfree*] to the freed block's number and increment *s_nfree*.

S_tfree is the total free blocks available in the file system.

S_ninode is the number of free i-numbers in the *s_inode* array. To allocate an i-node: if *s_ninode* is greater than 0, decrement it and return *s_inode*[*s_ninode*]. If it was 0, read the i-list and place the numbers of all free i-nodes (up to 100) into the *s_inode* array, then try again. To free an i-node, provided *s_ninode* is less than 100, place its number into *s_inode*[*s_ninode*] and increment *s_ninode*. If *s_ninode* is already 100, do not bother to enter the freed i-node into any table. This list of i-nodes is only to speed up the allocation process; the information as to whether the i-node is really free or not is maintained in the i-node itself.

S_tinode is the total free i-nodes available in the file system.

S_flock and *s_iloc* are flags maintained in the core copy of the file system while it is mounted and their values on disk are immaterial. The value of *s_fmmod* on disk is likewise immaterial; it is used as a flag to indicate that the super-block has changed and should be copied to the disk during the next periodic update of file system information.

S_ronly is a read-only flag to indicate write-protection.

S_time is the last time the super-block of the file system was changed, and is the number of seconds that have elapsed since 00:00 Jan. 1, 1970 (GMT). During a reboot, the *s_time* of the super-block for the root file system is used to set the system's idea of the time.

S_fname is the name of the file system and *s_fpack* is the name of the pack.

I-numbers begin at 1, and the storage for i-nodes begins in block 2. Also, i-nodes are 64 bytes long. I-node 1 is reserved for future use. I-node 2 is reserved for the root directory of the file system, but no other i-number has a built-in meaning. Each i-node represents one file. For the format of an i-node and its flags, see *inode(4)*.

SEE ALSO

mount(2) in the *Programmer's Reference Manual*.

fsck(1M), *fsdb(1M)*, *mkfs(1M)*, *inode(4)* in the *System Administrator's Reference Manual*.



NAME

`fspec` – format specification in text files

DESCRIPTION

It is sometimes convenient to maintain text files on the UNIX system with non-standard tabs, (i.e., tabs which 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 UNIX 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:

ttabs The **t** parameter specifies the tab settings for the file. The value of *ttabs* must be one of the following:

1. a list of column numbers separated by commas, indicating tabs set at the specified columns;
2. a `-` followed immediately by an integer *n*, indicating tabs at intervals of *n* columns;
3. 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 which are recognized are defined by the *tabs(1)* command.

ssize The **s** 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.

mmargin The **m** 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.

FSPEC(4)

FSPEC(4)

SEE ALSO

ed(1), newform(1), tabs(1) in the *User's Reference Manual*.



NAME

fstab – file-system-table

DESCRIPTION

The `/etc/fstab` file contains information about file systems for use by `mount(1M)` and `mountall(1M)`. Each entry in `/etc/fstab` has the following format:

column 1	block special file name of file system or advertised remote resource
column 2	mount-point directory
column 3	"-r" if to be mounted read-only; "-d[r]" if remote
column 4	(optional) file system type string
column 5+	ignored

White-space separates columns. Lines beginning with "# " are comments. Empty lines are ignored.

A file-system-table might read:

```
/dev/dsk/c1d0s2 /usr S51K
/dev/dsk/c1d1s2 /usr/src -r
adv_resource /mnt -d
```

FILES

`/etc/fstab`

SEE ALSO

`mount(1M)`, `mountall(1M)`, `rmountall(1M)` in the *System Administrator's Reference Manual*.



NAME

gettydefs – speed and terminal settings used by getty

DESCRIPTION

The `/etc/gettydefs` file contains information used by `getty(1M)` to set up the speed and terminal settings for a line. It supplies information on what the `login(1)` prompt should look like. It also supplies the speed to try next if the user indicates the current speed is not correct by typing a `<break>` character.

NOTE: Customers who need to support terminals that pass 8 bits to the system (as is typical outside the U.S.A.) must modify the entries in `/etc/gettydefs` as described in the WARNINGS section.

Each entry in `/etc/gettydefs` has the following format:

```
label# initial-flags # final-flags # login-prompt #next-label
```

Each entry is followed by a blank line. The various fields can contain quoted characters of the form `\b`, `\n`, `\c`, etc., as well as `\nnn`, where `nnn` is the octal value of the desired character. The various fields are:

- label* This is the string against which `getty(1M)` tries to match its second argument. It is often the speed, such as `1200`, at which the terminal is supposed to run, but it need not be (see below).
- initial-flags* These flags are the initial `ioctl(2)` settings to which the terminal is to be set if a terminal type is not specified to `getty(1M)`. The flags that `getty(1M)` understands are the same as the ones listed in `/usr/include/sys/termio.h` [see `termio(7)`]. Normally only the speed flag is required in the *initial-flags*. `getty(1M)` automatically sets the terminal to raw input mode and takes care of most of the other flags. The *initial-flag* settings remain in effect until `getty(1M)` executes `login(1)`.
- final-flags* These flags take the same values as the *initial-flags* and are set just before `getty(1M)` executes `login(1)`. The speed flag is again required. The composite flag `SANE` takes care of most of the other flags that need to be set so that the processor and terminal are communicating in a rational fashion. The other two commonly specified *final-flags* are `TAB3`, so that tabs are sent to the terminal as spaces, and `HUPCL`, so that the line is hung up on the final close.
- login-prompt* This entire field is printed as the *login-prompt*. Unlike the above fields where white space is ignored (a space, tab or new-line), they are included in the *login-prompt* field.
- next-label* If this entry does not specify the desired speed, indicated by the user typing a `<break>` character, then `getty(1M)` will search for the entry with *next-label* as its *label* field and set up the terminal for those settings. Usually, a series of speeds are linked together in this fashion, into a closed set; for instance, `2400` linked to `1200`, which in turn is linked to `300`, which finally is linked to `2400`.

If *getty(1M)* is called without a second argument, then the first entry of */etc/gettydefs* is used, thus making the first entry of */etc/gettydefs* the default entry. It is also used if *getty(1M)* can not find the specified *label*. If */etc/gettydefs* itself is missing, there is one entry built into *getty(1M)* which will bring up a terminal at 300 baud.

It is strongly recommended that after making or modifying */etc/gettydefs*, it be run through *getty(1M)* with the check option to be sure there are no errors.

FILES

/etc/gettydefs

SEE ALSO

getty(1M), *termio(7)* in the *System Administrator's Reference Manual*.

ioctl(2) in the *Programmer's Reference Manual*.

login(1), *stty(1)* in the *User's Reference Manual*.

WARNINGS

To support terminals that pass 8 bits to the system (also, see the BUGS section), modify the entries in the */etc/gettydefs* file for those terminals as follows: add *CS8* to *initial-flags* and replace all occurrences of *SANE* with the values: *BRKINT IGNPAR ICRNL IXON OPOST ONCLR CS8 ISIG ICANON ECHO ECHOK*

An example of changing an entry in */etc/gettydefs* is illustrated below. All the information for an entry must be on one line in the file.

Original entry:

```
CONSOLE # B9600 HUPCL OPOST ONLCR # B9600 SANE
IXANY TAB3 HUPCL # Console Login: # console
```

Modified entry:

```
CONSOLE # B9600 CS8 HUPCL OPOST ONLCR # B9600
BRKINT IGNPAR ICRNL IXON OPOST ONLCR CS8 ISIG
ICANON ECHO ECHOK IXANY TAB3 HUPCL # Console
Login: # console
```

This change will permit terminals to pass 8 bits to the system so long as the system is in *MULTI-USER* state. When the system changes to *SINGLE-USER* state, the *getty(1M)* is killed and the terminal attributes are lost. So to permit a terminal to pass 8 bits to the system in *SINGLE-USER* state, after you are in *SINGLE-USER* state, type (see *stty(1)*):

```
stty -istrip cs8
```

BUGS

8-bit with parity mode is not supported.

NAME

gps – graphical primitive string, format of graphical files

DESCRIPTION

GPS is a format used to store graphical data. Several routines have been developed to edit and display GPS files on various devices. Also, higher level graphics programs such as *plot* [in *stat(1G)*] and *vtoc* [in *toc(1G)*] produce GPS format output files.

A GPS is composed of five types of graphical data or primitives.

GPS PRIMITIVES

lines The *lines* primitive has a variable number of points from which zero or more connected line segments are produced. The first point given produces a *move* to that location. (A *move* is a relocation of the graphic cursor without drawing.) Successive points produce line segments from the previous point. Parameters are available to set *color*, *weight*, and *style* (see below).

arc The *arc* primitive has a variable number of points to which a curve is fit. The first point produces a *move* to that point. If only two points are included, a line connecting the points will result; if three points a circular arc through the points is drawn; and if more than three, lines connect the points. (In the future, a spline will be fit to the points if they number greater than three.) Parameters are available to set *color*, *weight*, and *style*.

text The *text* primitive draws characters. It requires a single point which locates the center of the first character to be drawn. Parameters are *color*, *font*, *textsize*, and *textangle*.

hardware The *hardware* primitive draws hardware characters or gives control commands to a hardware device. A single point locates the beginning location of the *hardware* string.

comment A *comment* is an integer string that is included in a GPS file but causes nothing to be displayed. All GPS files begin with a comment of zero length.

GPS PARAMETERS

color *Color* is an integer value set for *arc*, *lines*, and *text* primitives.

weight *Weight* is an integer value set for *arc* and *lines* primitives to indicate line thickness. The value 0 is narrow weight, 1 is bold, and 2 is medium weight.

style *Style* is an integer value set for *lines* and *arc* primitives to give one of the five different line styles that can be drawn on TEKTRONIX 4010 series storage tubes. They are:

- 0 solid
- 1 dotted
- 2 dot dashed
- 3 dashed
- 4 long dashed

- font** An integer value set for *text* primitives to designate the text font to be used in drawing a character string. (Currently *font* is expressed as a four-bit *weight* value followed by a four-bit *style* value.)
- textsize** *Textsize* is an integer value used in *text* primitives to express the size of the characters to be drawn. *Textsize* represents the height of characters in absolute *universe-units* and is stored at one-fifth this value in the size-orientation (*so*) word (see below).
- textangle** *Textangle* is a signed integer value used in *text* primitives to express rotation of the character string around the beginning point. *Textangle* is expressed in degrees from the positive x-axis and can be a positive or negative value. It is stored in the size-orientation (*so*) word as a value 256/360 of it's absolute value.

ORGANIZATION

GPS primitives are organized internally as follows:

lines	<i>cw points sw</i>
arc	<i>cw points sw</i>
text	<i>cw point sw so [string]</i>
hardware	<i>cw point [string]</i>
comment	<i>cw [string]</i>

- cw** *Cw* is the control word and begins all primitives. It consists of four bits that contain a primitive-type code and twelve bits that contain the word-count for that primitive.
- point(s)** *Point(s)* is one or more pairs of integer coordinates. *Text* and *hardware* primitives only require a single *point*. *Point(s)* are values within a Cartesian plane or *universe* having 64K (-32K to +32K) points on each axis.
- sw** *Sw* is the style-word and is used in *lines*, *arc*, and *text* primitives. For all three, eight bits contain *color* information. In *arc* and *lines* eight bits are divided as four bits *weight* and four bits *style*. In the *text* primitive eight bits of *sw* contain the *font*.
- so** *So* is the size-orientation word used in *text* primitives. Eight bits contain text size and eight bits contain text rotation.
- string** *String* is a null-terminated character string. If the string does not end on a word boundary, an additional null is added to the GPS file to insure word-boundary alignment.

SEE ALSO

graphics(1G), stat(1G), toc(1G) in the *User's Reference Manual*.

NAME

group – group file

DESCRIPTION

group contains for each group the following information:

- group name
- encrypted password
- numerical group ID
- comma-separated list of all users allowed in the group

This is an ASCII file. The fields are separated by colons; each group is separated from the next by a new-line. If the password field is null, no password is demanded.

This file resides in directory */etc*. Because of the encrypted passwords, it can and does have general read permission and can be used, for example, to map numerical group ID's to names.

FILES

/etc/group

SEE ALSO

newgrp(1M), *passwd(4)*.
passwd(1) in the *User's Reference Manual*.



NAME

`inittab` – script for the `init` process

DESCRIPTION

The *inittab* file supplies the script to *init*'s role as a general process dispatcher. The process that constitutes the majority of *init*'s process dispatching activities is the line process `/etc/getty` that initiates individual terminal lines. Other processes typically dispatched by *init* are daemons and the shell.

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 per entry are permitted. Comments may be inserted in the *process* field using the *sh*(1) convention for comments. Comments for lines that spawn *gettys* are displayed by the *who*(1) command. It is expected that they will contain some information about the line such as the location. There are no limits (other than maximum entry size) imposed on the number of entries within the *inittab* file. The entry fields are:

- id* This is one or two characters used to uniquely identify an entry.
- rstate* This defines 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* or *run-levels* in which it is allowed to exist. The *run-levels* are represented by a number ranging from 0 through 6. As an example, if the system is in *run-level* 1, only those entries having a 1 in the *rstate* field will be processed. When *init* is requested to change *run-levels*, all processes which do not have an entry in the *rstate* field for the target *run-level* will be sent the warning signal (SIGTERM) and allowed a 20-second grace period before being forcibly terminated by a 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 0–6. If no *run-level* is specified, then the process is assumed to be valid at all *run-levels* 0–6. 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 the *telinit* [see *init*(1M)] process requests them to be run (regardless of the current *run-level* of the system). They 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 only killed if their line in `/etc/inittab` is marked **off** in the *action* field, their line is deleted entirely from `/etc/inittab`, or *init* goes into the *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:

- respawn** If the process does not exist then start the process, do not wait for its termination (continue scanning the *inittab* file), and when it dies restart the process. If the process currently exists then do nothing and continue scanning the *inittab* file.
- wait** Upon *init*'s entering 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* will cause *init* to ignore this entry.
- once** Upon *init*'s entering 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 upon entering a new *run-level*, where the process is still running from a previous *run-level* change, the program will not be 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, not wait for its termination; and when it dies, 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 will run 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(2)*].
- 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 20 seconds before forcibly terminating the process via 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 is used only with the *a*, *b* or *c* values described in the *rstate* field.

initdefault An entry with this *action* is only scanned when *init* initially invoked. *Init* uses this entry, if it exists, 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 so *init* will enter *run-level* 6. Additionally, if *init* does not find an **initdefault** entry in */etc/inittab*, then it will request an initial *run-level* from the user at reboot time.

sysinit Entries of this type are executed before *init* tries to access the console (i.e., before the **Console Login:** prompt). It is expected that this entry will be only used to initialize devices on which *init* might try to ask the *run-level* question. These entries are executed and waited for before continuing.

process This is a *sh* 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. Comments can be inserted with the **;** *#comment* syntax.

FILES

/etc/inittab

SEE ALSO

exec(2), *open(2)*, *signal(2)* in the *Programmer's Reference Manual*.
getty(1M), *init(1M)* in the *System Administrator's Reference Manual*.
sh(1), *who(1)* in the *User's Reference Manual*.



NAME

inode – format of an i-node

SYNOPSIS

```
#include <sys/types.h>
#include <sys/ino.h>
```

DESCRIPTION

An i-node for a plain file or directory in a file system has the following structure defined by `<sys/ino.h>`.

```
/* Inode structure as it appears on a disk block. */
struct dinode
{
    ushort di_mode; /* mode and type of file */
    short di_nlink; /* number of links to file */
    ushort di_uid; /* owner's user id */
    ushort di_gid; /* owner's group id */
    off_t di_size; /* number of bytes in file */
    char di_addr[40]; /* disk block addresses */
    time_t di_atime; /* time last accessed */
    time_t di_mtime; /* time last modified */
    time_t di_ctime; /* time of last file status change */
};
/*
 * the 40 address bytes:
 * 39 used; 13 addresses
 * of 3 bytes each.
 */
```

For the meaning of the defined types `off_t` and `time_t` see `types(5)`.

SEE ALSO

`fs(4)`, `types(5)`.
`stat(2)` in the *Programmer's Reference Manual*.



NAME

issue – issue identification file

DESCRIPTION

The file `/etc/issue` contains the *issue* or project identification to be printed as a login prompt. This is an ASCII file which 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)` in the *User's Reference Manual*.



NAME

ldfcn – common object file access routines

SYNOPSIS

```
#include <stdio.h>
#include <filehdr.h>
#include <ldfcn.h>
```

DESCRIPTION

The common object file access routines are a collection of functions for reading common object files and archives containing common object files. Although the calling program must know the detailed structure of the parts of the object file that it processes, the routines effectively insulate the calling program from knowledge of the overall structure of the object file.

The interface between the calling program and the object file access routines is based on the defined type `LDFILE`, defined as `struct ldfile`, declared in the header file `ldfcn.h`. The primary purpose of this structure is to provide uniform access to both simple object files and to object files that are members of an archive file.

The function `ldopen(3X)` allocates and initializes the `LDFILE` structure and returns a pointer to the structure to the calling program. The fields of the `LDFILE` structure may be accessed individually through macros defined in `ldfcn.h` and contain the following information:

<code>LDFILE</code>	<code>*ldptr;</code>
<code>TYPE(ldptr)</code>	The file magic number used to distinguish between archive members and simple object files.
<code>IOPTR(ldptr)</code>	The file pointer returned by <code>fopen</code> and used by the standard input/output functions.
<code>OFFSET(ldptr)</code>	The file address of the beginning of the object file; the offset is non-zero if the object file is a member of an archive file.
<code>HEADER(ldptr)</code>	The file header structure of the object file.

The object file access functions themselves may be divided into four categories:

- (1) functions that open or close an object file

```
ldopen(3X) and ldaopen[see ldopen(3X)]
    open a common object file
ldclose(3X) and ldaclose[see ldclose(3X)]
    close a common object file
```

- (2) functions that read header or symbol table information

```
ldahread(3X)
    read the archive header of a member of an archive file
ldfhread(3X)
    read the file header of a common object file
ldshread(3X) and ldnshread[see ldshread(3X)]
    read a section header of a common object file
```

ldtbread(3X)
read a symbol table entry of a common object file

ldgetname(3X)
retrieve a symbol name from a symbol table entry or from the string table

(3) functions that position an object file at (seek to) the start of the section, relocation, or line number information for a particular section.

ldohseek(3X)
seek to the optional file header of a common object file

ldsseek(3X) and *ldnsseek*[see *ldsseek*(3X)]
seek to a section of a common object file

ldrseek(3X) and *ldnrseek*[see *ldrseek*(3X)]
seek to the relocation information for a section of a common object file

ldlseek(3X) and *ldnlseek*[see *ldlseek*(3X)]
seek to the line number information for a section of a common object file

ldtbseek(3X)
seek to the symbol table of a common object file

(4) the function *ldtbindx*(3X) which returns the index of a particular common object file symbol table entry.

These functions are described in detail on their respective manual pages.

All the functions except *ldopen*(3X), *ldgetname*(3X), *ldtbindx*(3X) return either SUCCESS or FAILURE, both constants defined in *ldfcn.h*. *Ldopen*(3X) and *ldaopen*[(see *ldopen*(3X))] both return pointers to an LDFILE structure.

Additional access to an object file is provided through a set of macros defined in *ldfcn.h*. These macros parallel the standard input/output file reading and manipulating functions, translating a reference of the LDFILE structure into a reference to its file descriptor field.

The following macros are provided:

GETC(ldptr)
FGETC(ldptr)
GETW(ldptr)
UNGETC(c, ldptr)
FGETS(s, n, ldptr)
FREAD((char *) ptr, sizeof (*ptr), nitems, ldptr)
FSEEK(ldptr, offset, ptrname)
FTELL(ldptr)
REWIND(ldptr)
FEOF(ldptr)
FERROR(ldptr)
FILENO(ldptr)
SETBUF(ldptr, buf)
STROFFSET(ldptr)

The STROFFSET macro calculates the address of the string table. See the manual entries for the corresponding standard input/output library functions for details on the use of the rest of the macros.

The program must be loaded with the object file access routine library **libld.a**.

SEE ALSO

intro(5).

fseek(3S), ldahread(3X), ldclose(3X), ldgetname(3X), ldhread(3X), ldhread(3X), ldlseek(3X), ldohseek(3X), ldopen(3X), ldrseek(3X), ldlseek(3X), ldshread(3X), ldtbindex(3X), ldtbread(3X), ldtbseek(3X), stdio(3S) in the *Programmer's Reference Manual*.

WARNING

The macro FSEEK defined in the header file **ldfcn.h** translates into a call to the standard input/output function *fseek(3S)*. FSEEK should not be used to seek from the end of an archive file since the end of an archive file may not be the same as the end of one of its object file members!



NAME

limits – file header for implementation-specific constants

SYNOPSIS

```
#include <limits.h>
```

DESCRIPTION

The header file *<limits.h>* is a list of magnitude limitations imposed by a specific implementation of the operating system. All values are specified in decimal.

```
#define ARG_MAX      5120 /* max length of arguments to exec */
#define CHAR_BIT     8   /* # of bits in a "char" */
#define CHAR_MAX     127 /* max integer value of a "char" */
#define CHAR_MIN     -128 /* min integer value of a "char" */
#define CHILD_MAX    25  /* max # of processes per user id */
#define CLK_TCK      100 /* # of clock ticks per second */
#define DBL_DIG      16  /* digits of precision of a "double" */
#define DBL_MAX      1.79769313486231470e+308 /*max decimal value of a "double"*/
#define DBL_MIN      4.94065645841246544e-324 /*min decimal value of a "double"*/
#define FCHR_MAX     1048576 /* max size of a file in bytes */
#define FLT_DIG      7   /* digits of precision of a "float" */
#define FLT_MAX      3.40282346638528860e+38 /*max decimal value of a "float" */
#define FLT_MIN      1.40129846432481707e-45 /*min decimal value of a "float" */
#define HUGE_VAL     3.40282346638528860e+38 /*error value returned by Math lib*/
#define INT_MAX      2147483647 /* max decimal value of an "int" */
#define INT_MIN      -2147483648 /* min decimal value of an "int" */
#define LINK_MAX     32767 /* max # of links to a single file */
#define LONG_MAX     2147483647 /* max decimal value of a "long" */
#define LONG_MIN     -2147483648 /* min decimal value of a "long" */
#define NAME_MAX     14   /* max # of characters in a file name */
#define OPEN_MAX     20   /* max # of files a process can have open */
#define PASS_MAX     8    /* max # of characters in a password */
#define PATH_MAX     256  /* max # of characters in a path name */
#define PID_MAX      30000 /* max value for a process ID */
#define PIPE_BUF     5120 /* max # bytes atomic in write to a pipe */
#define PIPE_MAX     5120 /* max # bytes written to a pipe in a write */
#define SHRT_MAX     32767 /* max decimal value of a "short" */
#define SHRT_MIN     -32767 /* min decimal value of a "short" */
#define STD_BLK      1024 /* # bytes in a physical I/O block */
#define SYS_NMLN     9    /* # of chars in uname-returned strings */
#define UID_MAX      30000 /* max value for a user or group ID */
#define USI_MAX      4294967296 /* max decimal value of an "unsigned" */
#define WORD_BIT     32   /* # of bits in a "word" or "int" */
```



NAME

linenum – line number entries in a common object file

SYNOPSIS

```
#include <linenum.h>
```

DESCRIPTION

The `cc` command generates an entry in the object file for each C source line on which a breakpoint is possible [when invoked with the `-g` option; see `cc(1)`]. Users can then reference line numbers when using the appropriate software test system [see `sdb(1)`]. The structure of these line number entries appears below.

```
struct lineno
{
    union
    {
        long    l_symndx ;
        long    l_paddr ;
    }          l_addr ;
    unsigned short l_lno ;
};
```

Numbering starts with one for each function. The initial line number entry for a function has `l_lno` equal to zero, and the symbol table index of the function's entry is in `l_symndx`. Otherwise, `l_lno` is non-zero, and `l_paddr` is the physical address of the code for the referenced line. Thus the overall structure is the following:

<i>l_addr</i>	<i>l_lno</i>
function symtab index	0
physical address	line
physical address	line
...	
function symtab index	0
physical address	line
physical address	line
...	

SEE ALSO

`cc(1)`, `sdb(1)`, `a.out(4)`.



NAME

/usr/adm/loginlog – log of failed login attempts

DESCRIPTION

After five unsuccessful login attempts, all the attempts are logged in the **loginlog** file. This file contains one record for each failed attempt. Each record contains the following information:

login name
tty specification
time

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.

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**.

FILES

/usr/adm/loginlog

SEE ALSO

login(1), **passwd(1)** in the *User's Reference Manual*.
passwd(1M) in the *System Administrator's Reference Manual*.



NAME

master – master configuration database

DESCRIPTION

The *master* configuration database is a collection of files. Each file contains configuration information for a device or module that may be included in the system. A file is named with the module name to which it applies. This collection of files is maintained in a directory called `/etc/master.d`. Each individual file has an identical format. For convenience, this collection of files will be referred to as the *master* file, as though it was a single file. This will allow a reference to the *master* file to be understood to mean the *individual file* in the `master.d` directory that corresponds to the name of a device or module. The file is used by the `mkboot(1M)` program to obtain device information to generate the device driver and configurable module files. It is also used by the `sysdef(1M)` program to obtain the names of supported devices. *master* consists of two parts; they are separated by a line with a dollar sign (\$) in column 1. Part 1 contains device information for both hardware and software devices, and loadable modules. Part 2 contains parameter declarations used in part 1. Any line with an asterisk (*) in column 1 is treated as a comment.

Part 1, Description

Hardware devices, software drivers and loadable modules are defined with a line containing the following information. Field 1 must begin in the left most position on the line. Fields are separated by white space (tab or blank).

Field 1:	element characteristics:
	o specify only once
	r required device
	b block device
	c character device
	a generate segment descriptor array
	t initialize <code>cdevsw[].d_ttys</code>
	s software driver
	f STREAMS driver
	m STREAMS module
	x not a driver; a loadable module
	number The first interrupt vector for an integral device
Field 2:	number of interrupt vectors required by a hardware device; "-" if none.
Field 3:	handler prefix (4 chars. maximum)
Field 4:	software driver external major number; "-" if not a software driver, or to be assigned during execution of <code>drvinstall(1M)</code>
Field 5:	number of sub-devices per device; "-" if none
Field 6:	interrupt priority level of the device; "-" if none
Field 7:	dependency list (optional); this is a comma separated list of other drivers or modules that must be present in the configuration if this module is to be included

For each module, two classes of information are required by *mkboot(1M)*: external routine references and variable definitions. Routine and variable definition lines begin with white space and immediately follow the initial module specification line. These lines are free form, thus they may be continued arbitrarily between non-blank tokens as long as the first character of a line is white space.

Part 1, Routine Reference Lines

If the UNIX system kernel or other dependent module contains external references to a module, but the module is not configured, then these external references would be undefined. Therefore, the *routine reference* lines are used to provide the information necessary to generate appropriate dummy functions at boot time when the driver is not loaded.

Routine references are defined as follows:

```
Field 1:  routine name ()
Field 2:  the routine type: one of
          {} routine_name(){}
          {nosys} routine_name(){return nosys();}
          {nodev} routine_name(){return nodev();}
          {false} routine_name(){return 0;}
          {true} routine_name(){return 1;}
```

Part 1, Variable Definition Lines

Variable definition lines are used to generate all variables required by the module. The variable generated may be of arbitrary size, be initialized or not, or be arrays containing an arbitrary number of elements.

variable references are defined as follows:

```
Field 1:  variable_name
Field 2:  [ expr ] – optional field used to indicate array size
Field 3:  (length) – required field indicating the size of the variable
Field 4:  = { expr,... } – optional field used to initialize individual elements of a variable
```

The *length* field is mandatory. It is an arbitrary sequence of length specifiers, each of which may be one of the following:

```
%i      an integer
%l      a long integer
%s      a short integer
%c      a single character
%number a field which is number bytes long
%number c a character string which is number bytes long
```

For example, the length field

```
( %8c %l %0x58 %l %c %c )
```

could be used to identify a variable consisting of a character string 8-bytes long, a long integer, a 0x58 byte structure of any type, another long integer, and two characters. Appropriate alignment of each % specification is performed (%number is word aligned) and the variable length is rounded up to the next word boundary during processing.

The expressions for the optional array size and initialization are infix expressions consisting of the usual operators for addition, subtraction, multiplication, and division: +, -, *, and /. Multiplication and division have the higher precedence, but parentheses may be used to override the default order. The builtin functions *min* and *max* accept a pair of expressions, and return the appropriate value. The operands of the expression may be any mixture of the following:

&name	address of name where <i>name</i> is any symbol defined by the kernel, any module loaded or any variable definition line of any module loaded
#name	sizeof name where <i>name</i> is any variable name defined by a variable definition for any module loaded; the size is that of the individual variable--not the size of an entire array
#C	number of controllers present; this number is determined by the EDT for hardware devices, or by the number provided in the system file for non-hardware drivers or modules
#C(name)	number of controllers present for the module <i>name</i> ; this number is determined by the EDT for hardware devices, or by the number provided in the system file for non-hardware drivers or modules
#D	number of devices per controller taken directly from the current master file entry
#D(name)	number of devices per controller taken directly from the master file entry for the module <i>name</i>
#M	the internal major number assigned to the current module if it is a device driver; zero if this module is not a device driver
#M(name)	the internal major number assigned to the module <i>name</i> if it is a device driver: zero if that module is not a device driver
name	value of a parameter as defined in the second part of <i>master</i>
number	arbitrary number (octal, decimal, or hex allowed)
string	a character string enclosed within double quotes (all of the character string conventions supported by the C language are allowed); this operand has a value which is the address of a character array containing the specified string

When initializing a variable, one initialization expression should be provided for each %i, %l, %s, or %c of the length field. The only initializers allowed for a '%number c' are either a character string (the string may not be longer than *number*), or an explicit zero. Initialization expressions must be separated by commas, and variable initialization will proceed element by element. Note that %number specification cannot be initialized--they are set to zero. Only the first element of an array can be initialized, the other elements are set to zero. If there are more initializers than size specifications, it is an error and execution of the *mkboot(1M)* program will be aborted. If there are fewer initializations than size specifications, zeros will be used to pad the variable. For example:

```
={"V2.L1", #C*#D, max(10,#D), #C(OTHER), #M(OTHER) }
```

would be a possible initialization of the variable whose length field was given in the preceding example.

Part 2, Description

Parameter declarations may be used to define a value symbolically. Values can be associated with identifiers and these identifiers may be used in the *variable definition* lines.

Parameters are defined as follows:

Field 1:	identifier (8 characters maximum)
Field 2:	=
Field 3:	value, the value may be a number (decimal, octal, or hex allowed), or a string

EXAMPLE

A sample *master* file for a tty device driver would be named "**atty**" if the device appeared in the EDT as "ATTY". The driver is a character device, the driver prefix is **at**, two interrupt vectors are used, and the interrupt priority is 6. In addition, another driver named "ATLOG" is necessary for the correct operation of the software associated with this device.

```
*FLAG #VEC PREFIX SOFT #DEV IPL DEPENDENCIES/VARIABLES
tca    2    at    -    2    6    ATLOG
                                atpoint(){false}
                                at_tty[#C*#D] (%0x58)
                                at_cnt(%i) = { #C*#D }
                                at_logmaj(%i) = { #M(ATLOG) }
                                at_id(%8c) = { ATID }
                                at_table(%i%1%31%s)
                                    = { max(#C,ATMAX),
                                        &at_tty,
                                        #C }

$
ATID = "fred"
ATMAX = 6
```

This *master* file will cause a routine named *atpoint* to be generated by the boot program if the ATTY driver is not loaded, and there is a reference to this routine from any other module loaded. When the driver is loaded, the variables *at_tty*, *at_cnt*, *at_logmaj*, *at_id*, and *at_table* will be allocated and initialized as specified. Due to the *t* flag, the *d_ttys* field in the character device switch table will be initialized to point to *at_tty* (the first variable definition line contains the variable whose address will be stored in *d_ttys*). The ATTY driver would reference these variables by coding:

```
extern struct tty at_tty[];
extern int at_cnt;
extern int at_logmaj;
extern char at_id[8];
extern struct {
    int member1;
    struct tty *member2;
    char junk[31];
    short member3;
} at_table;
```

FILES

/etc/master.d/*

SEE ALSO

drvinstall(1M), mkboot(1M), sysdef(1M), system(4).



NAME

`mnttab` – mounted file system table

SYNOPSIS

```
#include <mnttab.h>
```

DESCRIPTION

`mnttab` resides in directory `/etc` and contains a table of devices, mounted by the `mount(1M)` command, in the following structure as defined by `<mnttab.h>`:

```
struct mnttab {
    char    mt_dev[32];
    char    mt_filsys[32];
    short   mt_ro_flg;
    time_t  mt_time;
};
```

Each entry is 70 bytes in length; the first 32 bytes are the null-padded name of the place where the *special file* is mounted; the next 32 bytes represent the null-padded root name of the mounted special file; the remaining 6 bytes contain the mounted *special file's* read/write permissions and the date on which it was mounted.

The maximum number of entries in `mnttab` is based on the system parameter `NMOUNT` located in `/etc/master.d/kernel`, which defines the number of allowable mounted special files.

SEE ALSO

`mount(1M)`, `setmnt(1M)` in the *System Administrator's Reference Manual*.



NAME

.ott – files that hold object architecture information

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 UNIX 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.



NAME

passwd – password file

DESCRIPTION

/etc/passwd contains for each user the following information:

- login name
- password and (optional) aging
- numerical user ID
- numerical group ID
- GCOS job number, box number, optional GCOS user ID
- initial working directory
- program to use as shell

This is an ASCII file. Each field within each user's entry is separated from the next by a colon. The GCOS field is used only when communicating with that system, and in other installations can contain any desired information. Each user is separated from the next by a new-line. If the shell field is null, */bin/sh* is used.

This file has user login information, and has general read permission. It can therefore be used, for example, to map numerical user IDs to names.

The password field consists of the character *x* if there is a */etc/shadow* file. If */etc/shadow* does not exist, and the login does have a password, this field will contain an encrypted copy of the password. This field remains only for compatibility reasons when */etc/shadow* exists.

The encrypted password consists of 13 characters chosen from a 64-character alphabet (*., /, 0-9, A-Z, a-z*), except when the password is null, in which case the encrypted password is also null. Password aging is effected for a particular user if his encrypted password in the password file is followed by a comma and a non-null string of characters from the above alphabet. (Such a string must be introduced in the first instance by the super-user.)

The first character of the age, *M* say, denotes the maximum number of weeks for which a password is valid. A user who attempts to login after his password has expired will be forced to supply a new one. The next character, *m* say, denotes the minimum period in weeks that must expire before the password may be changed. The remaining one or two characters define the week (counted from the beginning of 1970) when the password was last changed. (A null string is equivalent to zero.) *M* and *m* have numerical values in the range 0-63 that correspond to the 64-character alphabet shown above (i.e., */* = 1 week; *z* = 63 weeks). If *m* = *M* = 0 (derived from the string *.* or *..*) the user will be forced to change his password the next time he logs in (and the "age" will disappear from his entry in the password file). If *m* > *M* (signified, e.g., by the string *./*) only the super-user will be able to change the password.

FILES

- /etc/passwd*
- /etc/shadow*

SEE ALSO

group(4).

getpwent(3C) in the *Programmer's Reference Manual*.

login(1), passwd(1) in the *User's Reference Manual*.

passwd(1M) in the *System Administrator's Reference Manual*.

NAME

plot – graphics interface

DESCRIPTION

Files of this format are produced by routines described in *plot(3X)* and are interpreted for various devices by commands described in *tplot(1G)*. 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.

Each of the following descriptions begins with the name of the corresponding routine in *plot(3X)*.

- 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 *tplot(1G)*].
- 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 new-line.
- e** erase: Start another frame of output.
- f** linemod: Take the following string, up to a new-line, as the style for drawing further lines. The styles are “dotted”, “solid”, “longdashed”, “shortdashed”, and “dotdashed”. Effective only for the **-T4014** and **-Tvers** options of *tplot(1G)* (TEKTRONIX 4014 terminal and Versatec plotter).
- 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 *tplot(1G)*. 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.

DASI 300	space(0, 0, 4096, 4096);
DASI 300s	space(0, 0, 4096, 4096);
DASI 450	space(0, 0, 4096, 4096);
TEKTRONIX 4014	space(0, 0, 3120, 3120);
Versatec plotter	space(0, 0, 2048, 2048);

SEE ALSO

gps(4), *term(5)*.
plot(3X) in the *Programmer's Reference Manual*.
graph(1G), *tplot(1G)* in the *User's Reference Manual*.

PLOT(4)

PLOT(4)

WARNING

The plotting library *plot(3X)* and the curses library *curses(3X)* both use the names *erase()* and *move()*. The curses versions are macros. If you need both libraries, put the *plot(3X)* code in a different source file than the *curses(3X)* code, and/or *#undef move()* and *erase()* in the *plot(3X)* code.



NAME

pnch – file format for card images

DESCRIPTION

The PNCH format is a convenient representation for files consisting of card images in an arbitrary code.

A PNCH file is a simple concatenation of card records. A card record consists of a single control byte followed by a variable number of data bytes. The control byte specifies the number (which must lie in the range 0-80) of data bytes that follow. The data bytes are 8-bit codes that constitute the card image. If there are fewer than 80 data bytes, it is understood that the remainder of the card image consists of trailing blanks.



NAME

profile – setting up an environment 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*(1) command. Computers running outside the Eastern time zone should have the line

```
. /etc/TIMEZONE
```

included early in */etc/profile* (see *timezone*(4)).

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 027
# Tell me when new mail comes in
MAIL=/usr/mail/$LOGNAME
# Add my /bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
# Set terminal type
while :
do
    echo "terminal: \c"
    read TERM
    if [ -f ${TERMINFO:-/usr/lib/terminfo}/?/$TERM ]
    then break
    elif [ -f /usr/lib/terminfo/?/$TERM ]
    then break
    else echo "invalid term $TERM" 1>&2
    fi
done
# Initialize the terminal and set tabs
# The environmental variable TERM must have been exported
# before the "tput init" command is executed.
tput init
# Set the erase character to backspace
stty erase '^H' echoe
```

FILES

/etc/TIMEZONEtimezone environment
\$HOME/.profile user-specific environment
/etc/profile system-wide environment

SEE ALSO

terminfo(4), timezone(4), environ(5), term(5).
env(1), login(1), mail(1), sh(1), stty(1), su(1), tput(1) in the *User's Reference Manual*.
su(1M) in the *System Administrator's Reference Manual*.
User's Guide.
Chapter 10 in the *Programmer'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.

NAME

reloc – relocation information for a common object file

SYNOPSIS

```
#include <reloc.h>
```

DESCRIPTION

Object files have one relocation entry for each relocatable reference in the text or data. If relocation information is present, it will be in the following format.

```
struct    reloc
{
    long    r_vaddr ; /* (virtual) address of reference */
    long    r_symndx ; /* index into symbol table */
    ushort  r_type ; /* relocation type */
};

#define R_ABS      0
#define R_DIR32   06
#define R_DIR32S  012
```

As the link editor reads each input section and performs relocation, the relocation entries are read. They direct how references found within the input section are treated.

R_ABS	The reference is absolute and no relocation is necessary. The entry will be ignored.
R_DIR32	A direct 32-bit reference to the symbol's virtual address.
R_DIR32S	A direct 32-bit reference to the symbol's virtual address, with the 32-bit value stored in the reverse order in the object file.

More relocation types exist for other processors. Equivalent relocation types on different processors have equal values and meanings. New relocation types will be defined (with new values) as they are needed.

Relocation entries are generated automatically by the assembler and automatically used by the link editor. Link editor options exist for both preserving and removing the relocation entries from object files.

SEE ALSO

as(1), ld(1), a.out(4), syms(4).



NAME

rfmaster – Remote File Sharing name server master file

DESCRIPTION

The **rfmaster** file is an ASCII file that identifies the hosts that are responsible for providing primary and secondary domain name service for Remote File Sharing domains. This file contains a series of records, each terminated by a newline; a record may be extended over more than one line by escaping the newline character with a backslash ("\n"). The fields in each record are separated by one or more tabs or spaces. Each record has three fields:

```
name type data
```

The type field, which defines the meaning of the *name* and *data* fields, has three possible values:

- p** The **p** type defines the primary domain name server. For this type, *name* is the domain name and *data* is the full host name of the machine that is the primary name server. The full host name is specified as *domain.nodename*. There can be only one primary name server per domain.
- s** The **s** type defines a secondary name server for a domain. *Name* and *data* are the same as for the **p** type. The order of the **s** entries in the **rfmaster** file determines the order in which secondary name servers take over when the current domain name server fails.
- a** The **a** type defines a network address for a machine. *Name* is the full domain name for the machine and *data* is the network address of the machine. The network address can be in plain ASCII text or it can be preceded by a \x to be interpreted as hexadecimal notation. (See the documentation for the particular network you are using to determine the network addresses you need.)

There are at least two lines in the **rfmaster** file per domain name server: one **p** and one **a** line, to define the primary and its network address. There should also be at least one secondary name server in each domain.

This file is created and maintained on the primary domain name server. When a machine other than the primary tries to start Remote File Sharing, this file is read to determine the address of the primary. If **rfmaster** is missing, the **-p** option of **rfstart** must be used to identify the primary. After that, a copy of the primary's **rfmaster** file is automatically placed on the machine.

Domains not served by the primary can also be listed in the **rfmaster** file. By adding primary, secondary, and address information for other domains on a network, machines served by the primary will be able to share resources with machines in other domains.

A primary name server may be a primary for more than one domain. However, the secondaries must then also be the same for each domain served by the primary.

EXAMPLES

An example of an **rfmaster** file is shown below. (The network address examples, *comp1.serve* and *comp2.serve*, are STARLAN network addresses.)

```
ccs          p      ccs.comp1
ccs          s      ccs.comp2
ccs.comp2   a      comp2.serve
ccs.comp1   a      comp1.serve
```

NOTE: If a line in the **rfmaster** file begins with a # character, the entire line will be treated as a comment.

FILES

/usr/nserve/rfmaster

SEE ALSO

rfstart(1M) in the *System Administrator's Reference Manual*.

NAME

sccsfile – format of SCCS file

DESCRIPTION

An SCCS (Source Code Control System) file is an ASCII file. It consists of six logical parts: the *checksum*, the *delta table* (contains information about each delta), *user names* (contains login names and/or numerical group IDs of users who may add deltas), *flags* (contains definitions of internal keywords), *comments* (contains arbitrary descriptive information about the file), and the *body* (contains the actual text lines intermixed with control lines).

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 graphically as @. Any line described below which is not depicted as beginning with the control character is prevented from beginning with the control character.

Entries of the form DDDDD represent a five-digit string (a number between 00000 and 99999).

Each logical part of an SCCS file is described in detail below.

Checksum

The checksum is the first line of an SCCS file. The form of the line is:
@hDDDDDD

The value of the checksum is the sum of all characters, except those of the first line. The @h provides a *magic number* of (octal) 064001.

Delta table

The delta table consists of a variable number of entries of the form:

```
@s DDDDD/DDDD/D
@d <type> <SCCS ID> yr/mo/da hr:mi:se <pgmr> DDDDD DDDDD
@i DDDDD ...
@x DDDDD ...
@g DDDDD ...
@m <MR number>
.
.
@c <comments> ...
.
.
@e
```

The first line (@s) contains the number of lines inserted/deleted/unchanged, respectively. The second line (@d) contains the type of the delta (currently, normal: D, and removed: R), the SCCS ID of the delta, the date and time of creation of the delta, the

login 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 @i, @x, and @g lines contain the serial numbers of deltas included, excluded, and ignored, respectively. These lines are optional.

The @m lines (optional) each contain one MR number associated with the delta; the @c lines contain comments associated with the delta.

The @e line ends the delta table entry.

User names

The list of login names and/or numerical group IDs of users who may add deltas to the file, separated by new-lines. The lines containing these login names and/or numerical group IDs are surrounded by the bracketing lines @u and @U. An empty list allows anyone to make a delta. Any line starting with a ! prohibits the succeeding group or user from making deltas.

Flags

Keywords used internally. [See *admin(1)* for more information on their use.] Each flag line takes the form:

```
@f <flag>    <optional text>
```

The following flags are defined:

```
@f t    <type of program>
@f v    <program name>
@f i    <keyword string>
@f b
@f m    <module name>
@f f    <floor>
@f c    <ceiling>
@f d    <default-sid>
@f n
@f j
@f l    <lock-releases>
@f q    <user defined>
@f z    <reserved for use in interfaces>
```

The t flag defines the replacement for the %Y% identification keyword. The v flag controls prompting for MR numbers in addition to comments; if the optional text is present it defines an MR number validity checking program. The i flag controls the warning/error aspect of the "No id keywords" message. When the i flag is not present, this message is only a warning; when the i flag is present, this message will cause a "fatal" error (the file will not be gotten, or the delta will not be made). When the b flag is present the -b keyletter may be used on

the *get* command to cause a branch in the delta tree. The *m* flag defines the first choice for the replacement text of the %M% identification keyword. The *f* flag defines the "floor" release; the release below which no deltas may be added. The *c* flag defines the "ceiling" release; the release above which no deltas may be added. The *d* flag defines the default SID to be used when none is specified on a *get* command. The *n* flag causes *delta* 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 (e.g., when delta 5.1 is made after delta 2.7, releases 3 and 4 are skipped). The absence of the *n* flag causes skipped releases to be completely empty. The *j* flag causes *get* to allow concurrent edits of the same base SID. The *l* flag defines a *list* of releases that are *locked* against editing [*get*(1) with the *-e* keyletter]. The *q* flag defines the replacement for the %Q% identification keyword. The *z* flag is used in certain specialized interface programs. *Comments* Arbitrary text is surrounded by the bracketing lines @t and @T. 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:

```
@I DDDDD
@D DDDDD
@E DDDDD
```

respectively. The digit string is the serial number corresponding to the delta for the control line.

SEE ALSO

admin(1), delta(1), get(1), prs(1) in the *Programmer's Reference Manual*.



NAME

scnhdr – section header for a common object file

SYNOPSIS

```
#include <scnhdr.h>
```

DESCRIPTION

Every common object file has a table of section headers to specify the layout of the data within the file. Each section within an object file has its own header. The C structure appears below.

```
struct scnhdr
{
    char          s_name[SYMNMLEN]; /* section name */
    long         s_paddr; /* physical address */
    long         s_vaddr; /* virtual address */
    long         s_size; /* section size */
    long         s_scnptr; /* file ptr to raw data */
    long         s_relptr; /* file ptr to relocation */
    long         s_lnnoptr; /* file ptr to line numbers */
    unsigned short s_nreloc; /* # reloc entries */
    unsigned short s_nlnno; /* # line number entries */
    long         s_flags; /* flags */
}
```

File pointers are byte offsets into the file; they can be used as the offset in a call to FSEEK [see *ldfcn(4)*]. If a section is initialized, the file contains the actual bytes. An uninitialized section is somewhat different. It has a size, symbols defined in it, and symbols that refer to it. But it can have no relocation entries, line numbers, or data. Consequently, an uninitialized section has no raw data in the object file, and the values for *s_scnptr*, *s_relptr*, *s_lnnoptr*, *s_nreloc*, and *s_nlnno* are zero.

SEE ALSO

ld(1), *fseek(3S)* in the *Programmer's Reference Manual*.
a.out(4).



NAME

`scr_dump` – format of curses screen image file.

SYNOPSIS

`scr_dump(file)`

DESCRIPTION

The `curses(3X)` function `scr_dump()` will copy the contents of the screen into a file. The format of the screen image is as described below.

The name of the tty is 20 characters long and the modification time (the *mtime* of the tty that this is an image of) is of the type *time_t*. All other numbers and characters are stored as *chtype* (see `<curses.h>`). No newlines are stored between fields.

```

    <magic number: octal 0433>
    <name of tty>
    <mod time of tty>
    <columns> <lines>
    <line length> <chars in line>      for each line on the screen
    <line length> <chars in line>
    .
    .
    .
    <labels?>                          1, if soft screen labels are present
    <cursor row> <cursor column>
  
```

Only as many characters as are in a line will be listed. For example, if the `<line length>` is 0, there will be no characters following `<line length>`. If `<labels?>` is TRUE, following it will be

```

    <number of labels>
    <label width>
    <chars in label 1>
    <chars in label 2>
    .
    .
    .
  
```

SEE ALSO

`curses(3X)` in the *Programmer's Reference Manual*.



NAME

syms – common object file symbol table format

SYNOPSIS

```
#include <syms.h>
```

DESCRIPTION

Common object files contain information to support symbolic software testing [see *sdb(1)*]. Line number entries, *linenum(4)*, and extensive symbolic information permit testing at the C *source* level. Every object file's symbol table is organized as shown below.

File name 1.

Function 1.

Local symbols for function 1.

Function 2.

Local symbols for function 2.

...

Static externs for file 1.

File name 2.

Function 1.

Local symbols for function 1.

Function 2.

Local symbols for function 2.

...

Static externs for file 2.

...

Defined global symbols.

Undefined global symbols.

The entry for a symbol is a fixed-length structure. The members of the structure hold the name (null padded), its value, and other information. The C structure is given below.

```
#define SYMNMLEN 8
#define FILNMLEN 14
#define DIMNUM 4
```

```
struct syment
```

```
{
    union /* all ways to get symbol name */
    {
        char _n_name[SYMNMLEN]; /* symbol name */
        struct
        {
            long _n_zeroes; /* == 0L when in string table */
            long _n_offset; /* location of name in table */
        } _n_n;
        char *_n_nptr[2]; /* allows overlaying */
    } _n;
}
```

```

long          n_value;    /* value of symbol */
short        n_scnum;    /* section number */
unsigned short n_type;   /* type and derived type */
char         n_sclass;   /* storage class */
char         n_numaux;   /* number of aux entries */
};

#define n_name      _n_n_name
#define n_zeroes    _n_n_n_n_zeroes
#define n_offset    _n_n_n_n_offset
#define n_nptr      _n_n_nptr[1]

```

Meaningful values and explanations for them are given in both `syms.h` and *Common Object File Format*. Anyone who needs to interpret the entries should seek more information in these sources. Some symbols require more information than a single entry; they are followed by *auxiliary entries* that are the same size as a symbol entry. The format follows.

```

union auxent
{
    struct
    {
        long          x_tagndx;
        union
        {
            struct
            {
                unsigned short x_inno;
                unsigned short x_size;
            } x_insz;
            long          x_fsize;
        } x_misc;
        union
        {
            struct
            {
                long          x_innoptr;
                long          x_endndx;
            } x_fcn;
            struct
            {
                unsigned short x_dimen[DIMNUM];
                x_ary;
                x_fcary;
            }
            unsigned short x_tvndx;
        }
    } x_sym;
    struct
    {
        char          x_fname[FILNMLEN];
        x_file;
    }
}

```

```
    struct
    {
        long    x_scrlen;
        unsigned short x_nreloc;
        unsigned short x_nlinno;
    } x_scn;

    struct
    {
        long          x_tvfill;
        unsigned short x_tvlen;
        unsigned short x_tvran[2];
    } x_tv;
};
```

Indexes of symbol table entries begin at *zero*.

SEE ALSO

sdb(1), a.out(4), linenum(4).

"Common Object File Format" in the *Programming Guide*.

WARNINGS

On machines on which **ints** are equivalent to **longs**, all **longs** have their type changed to **int**. Thus the information about which symbols are declared as **longs** and which, as **ints**, does not show up in the symbol table.



NAME

system – system configuration information table

DESCRIPTION

This file is used by the **boot** program to obtain configuration information that cannot be obtained from the equipped device table (EDT) at system boot time. This file generally contains a list of software drivers to include in the load, the assignment of system devices such as *pipdev* and *swapdev*, as well as instructions for manually overriding the drivers selected by the self-configuring boot process.

The syntax of the system file is given below. The parser for the */etc/system* file is case sensitive. All upper case strings in the syntax below should be upper case in the */etc/system* file as well. Nonterminal symbols are enclosed in angle brackets "<>" while optional arguments are enclosed in square brackets "[]". Ellipses "..." indicate optional repetition of the argument for that line.

```
<fname> ::= pathname
<string> ::= driver file name from /boot or EDT entry name
<device> ::= special device name | DEV(<major>,<minor>)
<major> ::= <number>
<minor> ::= <number>
<number> ::= decimal, octal or hex literal
```

The lines listed below may appear in any order. Blank lines may be inserted at any point. Comment lines must begin with an asterisk. Entries for EXCLUDE and INCLUDE are cumulative. For all other entries, the last line to appear in the file is used -- any earlier entries are ignored.

```
BOOT: <fname>
      specifies the kernel a.out file to be booted; if the file is
      fully resolved [such as that produced by the mkunix(1M)
      program] then all other lines in the system file have no
      effect.
EXCLUDE: [ <string> ] ...
      specifies drivers to exclude from the load even if the
      device is found in the EDT.
INCLUDE: [ <string>[( <number>)] ] ...
      specifies software drivers or loadable modules to be
      included in the load. This is necessary to include the
      drivers for software "devices". The optional <number>
      (parenthesis required) specifies the number of "devices"
      to be controlled by the driver (defaults to 1). This
      number corresponds to the builtin variable #c which
      may be referred to by expressions in part one of the
      /etc/master file.
ROOTDEV: <device>
      identifies the device containing the root file system.
SWAPDEV: <device> <number> <number>
      identifies the device to be used as swap space, the block
```

number the swap space starts at, and the number of swap blocks available.

PIPEDEV: <device>

identifies the device to be used for pipe space.

FILES

/etc/system

SEE ALSO

master(4).

crash(1M), mkunix(1M), mkboot(1M) in the *System Administrator's Reference Manual*.

NAME

term — format of compiled term file.

SYNOPSIS

```
/usr/lib/terminfo/?/*
```

DESCRIPTION

Compiled *terminfo(4)* descriptions are placed under the directory */usr/lib/terminfo*. In order to avoid a linear search of a huge UNIX system directory, a two-level scheme is used: */usr/lib/terminfo/c/name* where *name* is the name of the terminal, and *c* is the first character of *name*. Thus, **att4425** can be found in the file */usr/lib/terminfo/a/att4425*. Synonyms for the same terminal are implemented by multiple links to the same compiled file.

The format has been chosen so that it will be 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(4)* 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 $256 * \text{second} + \text{first}$.) The value -1 is represented by **0377,0377**, and the value -2 is represented by **0376,0377**; other negative values are illegal. Computers where this does not correspond to the hardware read the integers as two bytes and compute the result, making the compiled entries portable between machine types. The -1 generally means that a capability is missing from this terminal. The -2 means that the capability has been cancelled in the *terminfo(4)* 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(1M)*) by using the *terminfo(4)* compiler, *tic(1M)*, and read by the routine *setupterm()*. (See *curses(3X)*.) The file is divided into six parts: the header, terminal names, boolean flags, numbers, strings, and string table.

The header section begins the file. This section contains 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 names section comes next. It contains the first line of the *terminfo(4)* 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 boolean flags have 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>*.

Between the boolean section and the number section, a null byte will be inserted, if necessary, to ensure that the number section begins on an even byte. 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, an octal dump of the description for the AT&T Model 37 KSR is included:

```
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,

0000000 032 001      \0 032 \0 013 \0 021 001  3 \0  3  7  |  t
0000020  t  y  3  7  |  A  T  &  T      m  o  d  e  l
0000040  3  7      t  e  l  e  t  y  p  e \0 \0 \0 \0 \0
0000060 \0 \0 \0 001 \0 \0 \0 \0 \0 \0 \0 \0 001 \0 \0 \0
0000100 001 \0 \0 \0 \0 \0 377 377 377 377 377 377 377 377 377
0000120 377 377 377 377 377 377 377 377 377 377 377 377 377 & \0
0000140      \0 377 377 377 377 377 377 377 377 377 377 377 377
0000160 377 377  " \0 377 377 377 377  ( \0 377 377 377 377 377
0000200 377 377  0 \0 377 377 377 377 377 377 377 377  - \0 377 377
0000220 377 377 377 377 377 377 377 377 377 377 377 377 377
*
0000520 377 377 377 377 377 377 377 377 377 377 377 377 377  $ \0
0000540 377 377 377 377 377 377 377 377 377 377 377 377 377  * \0
0000560 377 377 377 377 377 377 377 377 377 377 377 377 377
*
0001160 377 377 377 377 377 377 377 377 377 377 377 377 377  3  7
0001200  |  t  t  y  3  7  |  A  T  &  T      m  o  d  e
0001220  1      3  7      t  e  l  e  t  y  p  e \0 \r \0
0001240 \n \0 \n \0 007 \0 \b \0 033  8 \0 033  9 \0 033  7
0001260 \0 \0
0001261
```

Some limitations: total compiled entries cannot exceed 4096 bytes; all entries in the name field cannot exceed 128 bytes.

FILES

/usr/lib/terminfo/??/* compiled terminal description database
/usr/include/term.h *terminfo(4)* header file

SEE ALSO

curses(3X) in the *Programmer's Reference Manual*.
infocmp(1M), *terminfo(4)*, *term(5)* in the *System Administrator's Reference Manual*.
Chapter 10 of the *Programmer's Guide*.



NAME

terminfo – terminal capability data base

SYNOPSIS

```
/usr/lib/terminfo/?/*
```

DESCRIPTION

terminfo is a database, produced by *tic*(1M), 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 effect particular results. This database is often used by screen oriented applications such as *vi*(1) and *curses*(3X), as well as by some UNIX system commands such as *ls*(1) and *pg*(1). This usage allows them to work with a variety of devices without changes to the programs. To obtain the source description for a device, use the *infocmp*(1M) command.

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. The following example shows the format of a *terminfo* source file.

Column 1



```
alias1 | alias2 | ... | aliasn | longname,  
<white space> am, lines #24,  
<white space> home=Eeh,
```

The first line, commonly referred to as the header line, must begin in column one and it 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 UNIX 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 "att5425" (for the AT&T model 5425 device). 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 an AT&T 5425 device set to wide mode, name the device "att5410-w." Use the following suffixes where possible.

Suffix	Meaning	Example
-w	Wide mode (more than 80 columns)	5410-w
-am	With auto. margins (usually default)	vt100-am
-nam	Without automatic margins	vt100-nam
-n	Number of lines on the screen	2300-40
-na	No arrow keys (leave them in local)	c100-na
-np	Number of pages of memory	c100-4p
-rv	Reverse video	4415-rv

The *terminfo* reference manual page is organized in two sections: "DEVICE CAPABILITIES" and "PRINTER CAPABILITIES."

PART 1: DEVICE 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 tables, 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(1)* 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 hard 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**_{*i*}. The **#i** symbol in the description field of the following tables refers to the *i*th parameter.

Booleans

Variable	Cap-name	Termcap Code	Description	Page#
auto_left_margin	bw	bw	cu b1 wraps from column 0 to last column	23
auto_right_margin	am	am	Terminal has automatic margins	24
back_color_erase	bce	be	Screen erased with background color	35

Variable	Cap-name	Termcap Code	Description	Page#
can_change	ccc	cc	Terminal can re-define existing color	34
ceol_standout_glitch	xhp	xs	Standout not erased by overwriting (hp)	37
col_addr_glitch	xhpa	YA	Only positive motion for hpa/mhpa caps	41
cpi_changes_res	cpix	YF	Changing character pitch changes resolution	39
cr_cancels_micro_mode	crxm	YB	Using cr turns off micro mode	42
eat_newline_glitch	xenl	xn	Newline ignored after 80 columns (<i>Concept</i>)	37
erase_overstrike	eo	eo	Can erase overstrikes with a blank	29
generic_type	gn	gn	Generic line type (e.g., dialup, switch)	36
hard_copy	hc	hc	Hardcopy terminal	23
hard_cursor	chts	HC	Cursor is hard to see	29
has_meta_key	km	km	Has a meta key (shift, sets parity bit)	36
has_print_wheel	daisy	YC	Printer needs operator to change character set	45
has_status_line	hs	hs	Has extra "status line"	32
hue_lightness_saturation	hls	hl	Terminal uses only HLS color notation (Tektronix)	34
insert_null_glitch	in	in	Insert mode distinguishes nulls	27
lpi_changes_res	lpix	YG	Changing line pitch changes resolution	39
memory_above	da	da	Display may be retained above the screen	27
memory_below	db	db	Display may be retained below the screen	27
move_insert_mode	mir	mi	Safe to move while in insert mode	28
move_standout_mode	msgr	ms	Safe to move in standout modes	29
needs_xon_xoff	nxon	nx	Padding won't work, xon/xoff required	
no_esc_ctlc	xsb	xb	Beehive (f1=escape, f2=ctrl C)	37
non_rev_rmcup	nrrmc	NR	smcup does not reverse rmcup	26
no_pad_char	npc	NP	Pad character doesn't exist	35
over_strike	os	os	Terminal overstrikes on hard-copy terminal	23
prtr_silent	mc5i	5i	Printer won't echo on screen	36
row_addr_glitch	xvpa	YD	Only positive motion for vpa/mvpa caps	41
semi_auto_right_margin	sam	YE	Printing in last column causes cr	42
status_line_esc_ok	eslok	es	Escape can be used on the status line	32
dest_tabs_magic_smo	xt	xt	Destructive tabs, magic smso char (t1061)	37
tilde_glitch	hz	hz	Hazeltine; can't print tilde (~)	37
transparent_underline	ul	ul	Underline character overstrikes	29
xon_xoff	xon	xo	Terminal uses xon/xoff handshaking	23

Numbers

Variable	Cap-name	Termcap Code	Description	Page#
buffer_capacity	bufsz	Ya	Number of bytes buffered before printing	50
columns	cols	co	Number of columns in a line	23
dot_vert_spacing	spinv	Yb	Spacing of pins vertically in pins per inch	47
dot_horz_spacing	spinh	Yc	Spacing of dots horizontally in dots per inch	47
init_tabs	it	it	Tabs initially every # spaces	31
label_height	lh	lh	Number of rows in each label	31

Variable	Cap-name	Termcap Code	Description	Page#
label_width	lw	lw	Number of columns in each label	31
lines	lines	li	Number of lines on a screen or a page	23
lines_of_memory	lm	lm	Lines of memory if > lines; 0 means varies	36
magic_cookie_glitch	xmc	sg	Number of blank characters left by smso or rmso	29
max_colors	colors	Co	Maximum number of colors on the screen	34
max_micro_address	maddr	Yd	Maximum value in micro_..._address	41
max_micro_jump	mjump	Ye	Maximum value in parm_..._micro	41
max_pairs	pairs	pa	Maximum number of color-pairs on the screen	34
micro_col_size	mcs	Yf	Character step size when in micro mode	39
micro_line_size	mls	Yg	Line step size when in micro mode	39
no_color_video	ncv	NC	Video attributes that can't be used with colors	35
number_of_pins	npins	Yh	Number of pins in print-head	47
num_labels	nlab	Nl	Number of labels on screen (start at 1)	31
output_res_char	orc	Yi	Horizontal resolution in units per character	38
output_res_line	orl	Yj	Vertical resolution in units per line	38
output_res_horz_inch	orhi	Yk	Horizontal resolution in units per inch	38
output_res_vert_inch	orvi	Yl	Vertical resolution in units per inch	38
padding_baud_rate	pb	pb	Lowest baud rate where padding needed	32
virtual_terminal	vt	vt	Virtual terminal number (UNIX system)	
wide_char_size	widcs	Yn	Character step size when in double wide mode	39
width_status_line	wsl	ws	Number of columns in status line	33

Strings

Variable	Cap-name	Termcap Code	Description	Page#
acs_chars	acsc	ac	Graphic charset pairs aAbBcC - def=vt100	33
back_tab	cbt	bt	Back tab	31
bell	bel	bl	Audible signal (bell)	23
carriage_return	cr	cr	Carriage return	23
change_char_pitch	cpi	ZA	Change number of characters per inch	39
change_line_pitch	lpi	ZB	Change number of lines per inch	39
change_res_horz	chr	ZC	Change horizontal resolution	39
change_res_vert	cvr	ZD	Change vertical resolution	39
change_scroll_region	csr	cs	Change to lines #1 through #2 (vt100)	26
char_padding	rmp	rP	Like ip but when in replace mode	28
char_set_names	csnm	Zy	List of character set names	45
clear_all_tabs	tbc	ct	Clear all tab stops	31
clear_margins	mgc	MC	Clear all margins (top, bottom, and sides)	32
clear_screen	clear	cl	Clear screen and home cursor	23
clr_bol	ell	cb	Clear to beginning of line, inclusive	
clr_eol	el	ce	Clear to end of line	26
clr_eos	ed	cd	Clear to end of display	26
column_address	hpa	ch	Horizontal position absolute	26

Variable	Cap-name	Termcap Code	Description	Page#
command_character	cmdch	CC	Terminal settable cmd character in prototype	36
cursor_address	cup	cm	Move to row #1 col #2	24
cursor_down	cud1	do	Down one line	23
cursor_home	home	ho	Home cursor (if no cup)	26
cursor_invisible	civis	vi	Make cursor invisible	29
cursor_left	cub1	le	Move left one space.	23
cursor_mem_address	mrcup	CM	Memory relative cursor addressing	24
cursor_normal	cnorm	ve	Make cursor appear normal (undo vs/vi)	29
cursor_right	cuf1	nd	Non-destructive space (cursor or carriage right)	23
cursor_to_ll	ll	ll	Last line, first column (if no cup)	26
cursor_up	cuu1	up	Upline (cursor up)	23
cursor_visible	cvvis	vs	Make cursor very visible	29
define_char	defc	ZE	Define a character in a character set †	45
delete_character	dch1	dc	Delete character	28
delete_line	dll	dl	Delete line	26
dis_status_line	dsl	ds	Disable status line	32
down_half_line	hd	hd	Half-line down (forward 1/2 linefeed)	35
ena_acs	enacs	eA	Enable alternate character set	28
enter_alt_charset_mode	smacs	as	Start alternate character set	28
enter_am_mode	smam	SA	Turn on automatic margins	
enter_blink_mode	blink	mb	Turn on blinking	28
enter_bold_mode	bold	md	Turn on bold (extra bright) mode	28
enter_ca_mode	smcup	ti	String to begin programs that use cup	26
enter_delete_mode	smdc	dm	Delete mode (enter)	28
enter_dim_mode	dim	mh	Turn on half-bright mode	28
enter_doublewide_mode	swidm	ZF	Enable double wide printing	44
enter_draft_quality	sdrfq	ZG	Set draft quality print	49
enter_insert_mode	smir	im	Insert mode (enter)	28
enter_italics_mode	sitm	ZH	Enable italics	44
enter_leftward_mode	slm	ZI	Enable leftward carriage motion	42
enter_micro_mode	smicm	ZJ	Enable micro motion capabilities	42
enter_near_letter_quality	snlq	ZK	Set near-letter quality print	49
enter_normal_quality	snrmq	ZL	Set normal quality print	49
enter_protected_mode	prot	mp	Turn on protected mode	28
enter_reverse_mode	rev	mr	Turn on reverse video mode	28
enter_secure_mode	invis	mk	Turn on blank mode (characters invisible)	
enter_shadow_mode	sshm	ZM	Enable shadow printing	44
enter_standout_mode	smso	so	Begin standout mode	29
enter_subscript_mode	ssubm	ZN	Enable subscript printing	44
enter_superscript_mode	ssupm	ZO	Enable superscript printing	44
enter_underline_mode	smul	us	Start underscore mode	29
enter_upward_mode	sum	ZP	Enable upward carriage motion	42
enter_xon_mode	smxon	SX	Turn on xon/xoff handshaking	36
erase_chars	ech	ec	Erase #1 characters	28
exit_alt_charset_mode	rmacs	ae	End alternate character set	

Variable	Cap-name	Termcap Code	Description	Page#
exit_am_mode	rmam	RA	Turn off automatic margins	
exit_attribute_mode	sgr0	me	Turn off all attributes	28
exit_ca_mode	rmcup	te	String to end programs that use cup	26
exit_delete_mode	rmdc	ed	End delete mode	28
exit_doublewide_mode	rwidm	ZQ	Disable double wide printing	44
exit_insert_mode	rmir	ei	End insert mode	28
exit_italics_mode	ritm	ZR	Disable italics	44
exit_leftward_mode	rlm	ZS	Enable rightward (normal) carriage motion	42
exit_micro_mode	rmicm	ZT	Disable micro motion capabilities	42
exit_shadow_mode	rshm	ZU	Disable shadow printing.	44
exit_standout_mode	rmso	se	End standout mode	29
exit_subscript_mode	rsubm	ZV	Disable subscript printing	44
exit_superscript_mode	rsupm	ZW	Disable superscript printing	44
exit_underline_mode	rmul	ue	End underscore mode	29
exit_upward_mode	rum	ZX	Enable downward (normal) carriage motion	42
exit_xon_mode	rmxon	RX	Turn off xon/xoff handshaking	36
flash_screen	flash	vb	Visible bell (may not move cursor)	29
form_feed	ff	ff	Hardcopy terminal page eject	32
from_status_line	fsl	fs	Return from status line	32
init_1string	is1	i1	Terminal or printer initialization string	31
init_2string	is2	is	Terminal or printer initialization string	31
init_3string	is3	i3	Terminal or printer initialization string	31
init_file	if	if	Name of initialization file	24
init_prog	ipro	iP	Path name of program for initialization	31
initialize_color	initc	Ic	Initialize the definition of color	34
initialize_pair	initp	Ip	Initialize color-pair	34
insert_character	ich1	ic	Insert character	28
insert_line	ill	al	Add new blank line	26
insert_padding	ip	ip	Insert pad after character inserted	28
key_a1	ka1	K1	KEY_A1, 0534, upper left of keypad	31
key_a3	ka3	K3	KEY_A3, 0535, upper right of keypad	31
key_b2	kb2	K2	KEY_B2, 0536, center of keypad	31
key_backspace	kbs	kb	KEY_BACKSPACE, 0407, sent by backspace key	31
key_beg	kbeg	@1	KEY_BEG, 0542, sent by beg(inning) key	
key_btab	kcbt	kB	KEY_BTAB, 0541, sent by back-tab key	
key_c1	kc1	K4	KEY_C1, 0537, lower left of keypad	31
key_c3	kc3	K5	KEY_C3, 0540, lower right of keypad	31
key_cancel	kcan	@2	KEY_CANCEL, 0543, sent by cancel key	
key_catab	ktbc	ka	KEY_CATAB, 0526, sent by clear-all-tabs key	31
key_clear	kclr	kC	KEY_CLEAR, 0515, sent by clear-screen or erase key	31
key_close	kclo	@3	KEY_CLOSE, 0544, sent by close key	
key_command	kcmd	@4	KEY_COMMAND, 0545, sent by cmd (command) key	
key_copy	kcpy	@5	KEY_COPY, 0546, sent by copy key	
key_create	kcrt	@6	KEY_CREATE, 0547, sent by create key	
key_ctab	kctab	kt	KEY_CTAB, 0525, sent by clear-tab key	31

Variable	Cap-name	Termcap Code	Description	Page#
key_dc	kdch1	kD	KEY_DC, 0512, sent by delete-character key	31
key_dl	kd11	kL	KEY_DL, 0510, sent by delete-line key	31
key_down	kcud1	kd	KEY_DOWN, 0402, sent by terminal down-arrow key	31
key_eic	krmir	kM	KEY_EIC, 0514, sent by rmir or smir in insert mode	31
key_end	kend	@7	KEY_END, 0550, sent by end key	
key_enter	kent	@8	KEY_ENTER, 0527, sent by enter/send key	
key_eol	kel	kE	KEY_EOL, 0517, sent by clear-to-end-of-line key	31
key_eos	ked	kS	KEY_EOS, 0516, sent by clear-to-end-of-screen key	31
key_exit	kext	@9	KEY_EXIT, 0551, sent by exit key	
key_f0	kf0	k0	KEY_F(0), 0410, sent by function key f0	31
key_f1	kf1	k1	KEY_F(1), 0411, sent by function key f1	31
key_f2	kf2	k2	KEY_F(2), 0412, sent by function key f2	31
key_f3	kf3	k3	KEY_F(3), 0413, sent by function key f3	31
key_f4	kf4	k4	KEY_F(4), 0414, sent by function key f4	31
key_f5	kf5	k5	KEY_F(5), 0415, sent by function key f5	31
key_f6	kf6	k6	KEY_F(6), 0416, sent by function key f6	31
key_f7	kf7	k7	KEY_F(7), 0417, sent by function key f7	31
key_f8	kf8	k8	KEY_F(8), 0420, sent by function key f8	31
key_f9	kf9	k9	KEY_F(9), 0421, sent by function key f9	31
key_f10	kf10	k;	KEY_F(10), 0422, sent by function key f10	31
key_f11	kf11	F1	KEY_F(11), 0423, sent by function key f11	31
key_f12	kf12	F2	KEY_F(12), 0424, sent by function key f12	31
key_f13	kf13	F3	KEY_F(13), 0425, sent by function key f13	31
key_f14	kf14	F4	KEY_F(14), 0426, sent by function key f14	31
key_f15	kf15	F5	KEY_F(15), 0427, sent by function key f15	31
key_f16	kf16	F6	KEY_F(16), 0430, sent by function key f16	31
key_f17	kf17	F7	KEY_F(17), 0431, sent by function key f17	31
key_f18	kf18	F8	KEY_F(18), 0432, sent by function key f18	31
key_f19	kf19	F9	KEY_F(19), 0433, sent by function key f19	31
key_f20	kf20	FA	KEY_F(20), 0434, sent by function key f20	31
key_f21	kf21	FB	KEY_F(21), 0435, sent by function key f21	31
key_f22	kf22	FC	KEY_F(22), 0436, sent by function key f22	31
key_f23	kf23	FD	KEY_F(23), 0437, sent by function key f23	31
key_f24	kf24	FE	KEY_F(24), 0440, sent by function key f24	31
key_f25	kf25	FF	KEY_F(25), 0441, sent by function key f25	31
key_f26	kf26	FG	KEY_F(26), 0442, sent by function key f26	31
key_f27	kf27	FH	KEY_F(27), 0443, sent by function key f27	31
key_f28	kf28	FI	KEY_F(28), 0444, sent by function key f28	31
key_f29	kf29	FJ	KEY_F(29), 0445, sent by function key f29	31
key_f30	kf30	FK	KEY_F(30), 0446, sent by function key f30	31
key_f31	kf31	FL	KEY_F(31), 0447, sent by function key f31	31
key_f32	kf32	FM	KEY_F(32), 0450, sent by function key f32	31
key_f33	kf33	FN	KEY_F(13), 0451, sent by function key f13	31
key_f34	kf34	FO	KEY_F(34), 0452, sent by function key f34	31
key_f35	kf35	FP	KEY_F(35), 0453, sent by function key f35	31

Variable	Cap-name	Termcap Code	Description	Page#
key_f36	kf36	FQ	KEY_F(36), 0454, sent by function key f36	31
key_f37	kf37	FR	KEY_F(37), 0455, sent by function key f37	31
key_f38	kf38	FS	KEY_F(38), 0456, sent by function key f38	31
key_f39	kf39	FT	KEY_F(39), 0457, sent by function key f39	31
key_f40	kf40	FU	KEY_F(40), 0460, sent by function key f40	31
key_f41	kf41	FV	KEY_F(41), 0461, sent by function key f41	31
key_f42	kf42	FW	KEY_F(42), 0462, sent by function key f42	31
key_f43	kf43	FX	KEY_F(43), 0463, sent by function key f43	31
key_f44	kf44	FY	KEY_F(44), 0464, sent by function key f44	31
key_f45	kf45	FZ	KEY_F(45), 0465, sent by function key f45	31
key_f46	kf46	Fa	KEY_F(46), 0466, sent by function key f46	31
key_f47	kf47	Fb	KEY_F(47), 0467, sent by function key f47	31
key_f48	kf48	Fc	KEY_F(48), 0470, sent by function key f48	31
key_f49	kf49	Fd	KEY_F(49), 0471, sent by function key f49	31
key_f50	kf50	Fe	KEY_F(50), 0472, sent by function key f50	31
key_f51	kf51	Ff	KEY_F(51), 0473, sent by function key f51	31
key_f52	kf52	Fg	KEY_F(52), 0474, sent by function key f52	31
key_f53	kf53	Fh	KEY_F(53), 0475, sent by function key f53	31
key_f54	kf54	Fi	KEY_F(54), 0476, sent by function key f54	31
key_f55	kf55	Fj	KEY_F(55), 0477, sent by function key f55	31
key_f56	kf56	Fk	KEY_F(56), 0500, sent by function key f56	31
key_f57	kf57	Fl	KEY_F(57), 0501, sent by function key f57	31
key_f58	kf58	Fm	KEY_F(58), 0502, sent by function key f58	31
key_f59	kf59	Fn	KEY_F(59), 0503, sent by function key f59	31
key_f60	kf60	Fo	KEY_F(60), 0504, sent by function key f60	31
key_f61	kf61	Fp	KEY_F(61), 0505, sent by function key f61	31
key_f62	kf62	Fq	KEY_F(62), 0506, sent by function key f62	31
key_f63	kf63	Fr	KEY_F(63), 0507, sent by function key f63	31
key_find	kfind	@0	KEY_FIND, 0552, sent by find key	
key_help	khlp	%1	KEY_HELP, 0553, sent by help key	
key_home	khome	kh	KEY_HOME, 0406, sent by home key	31
key_ic	kich1	kl	KEY_IC, 0513, sent by ins-char/enter ins-mode key	31
key_il	kil1	kA	KEY_IL, 0511, sent by insert-line key	31
key_left	kcub1	kl	KEY_LEFT, 0404, sent by terminal left-arrow key	31
key_ll	kl1	kH	KEY_LL, 0533, sent by home-down key	31
key_mark	kmrk	%2	KEY_MARK, 0554, sent by mark key	
key_message	kmsg	%3	KEY_MESSAGE, 0555, sent by message key	
key_move	kmov	%4	KEY_MOVE, 0556, sent by move key	
key_next	knxt	%5	KEY_NEXT, 0557, sent by next-object key	
key_npage	knp	kN	KEY_NPAGE, 0522, sent by next-page key	31
key_open	kopn	%6	KEY_OPEN, 0560, sent by open key	
key_options	kopt	%7	KEY_OPTIONS, 0561, sent by options key	
key_ppage	kpp	kP	KEY_PPAGE, 0523, sent by previous-page key	31
key_previous	kprv	%8	KEY_PREVIOUS, 0562, sent by previous-object key	
key_print	kprt	%9	KEY_PRINT, 0532, sent by print or copy key	

Variable	Cap-name	Termcap Code	Description	Page#
key_redo	krdo	%0	KEY_REDO, 0563, sent by redo key	
key_reference	kref	&1	KEY_REFERENCE, 0564, sent by ref(erence) key	
key_refresh	krfr	&2	KEY_REFRESH, 0565, sent by refresh key	
key_replace	krpl	&3	KEY_REPLACE, 0566, sent by replace key	
key_restart	krst	&4	KEY_RESTART, 0567, sent by restart key	
key_resume	kres	&5	KEY_RESUME, 0570, sent by resume key	
key_right	kcuf1	kr	KEY_RIGHT, 0405, sent by terminal right-arrow key	31
key_save	ksav	&6	KEY_SAVE, 0571, sent by save key	
key_sbeg	kBEG	&9	KEY_SBEG, 0572, sent by shifted beginning key	
key_scancel	kCAN	&0	KEY_SCANCEL, 0573, sent by shifted cancel key	
key_scommand	kCMD	*1	KEY_SCOMMAND, 0574, sent by shifted command key	
key_scopy	kCPY	*2	KEY_SCOPY, 0575, sent by shifted copy key	
key_screate	kCRT	*3	KEY_SCREATE, 0576, sent by shifted create key	
key_sdc	kDC	*4	KEY_SDC, 0577, sent by shifted delete-char key	
key_sdl	kDL	*5	KEY_SDL, 0600, sent by shifted delete-line key	
key_select	kslt	*6	KEY_SELECT, 0601, sent by select key	
key_send	kEND	*7	KEY_SEND, 0602, sent by shifted end key	
key_seol	kEOL	*8	KEY_SEOL, 0603, sent by shifted clear-line key	
key_sexit	kEXT	*9	KEY_SEXIT, 0604, sent by shifted exit key	
key_sf	kind	kF	KEY_SF, 0520, sent by scroll-forward/down key	31
key_sfind	kFND	*0	KEY_SFIND, 0605, sent by shifted find key	
key_shelp	kHLP	#1	KEY_SHELP, 0606, sent by shifted help key	
key_shome	kHOM	#2	KEY_SHOME, 0607, sent by shifted home key	
key_sic	kIC	#3	KEY_SIC, 0610, sent by shifted input key	
key_sleft	kLFT	#4	KEY_SLEFT, 0611, sent by shifted left-arrow key	
key_smessage	kMSG	%a	KEY_SMESSAGE, 0612, sent by shifted message key	
key_smove	kMOV	%b	KEY_SMOVE, 0613, sent by shifted move key	
key_snext	kNXT	%c	KEY_SNEXT, 0614, sent by shifted next key	
key_soptions	kOPT	%d	KEY_SOPTIONS, 0615, sent by shifted options key	
key_sprevious	kPRV	%e	KEY_SPREVIOUS, 0616, sent by shifted prev key	
key_sprint	kPRT	%f	KEY_SPRINT, 0617, sent by shifted print key	
key_sr	kri	kR	KEY_SR, 0521, sent by scroll-backward/up key	31
key_sredo	krDO	%g	KEY_SREDO, 0620, sent by shifted redo key	
key_sreplace	krPL	%h	KEY_SREPLACE, 0621, sent by shifted replace key	
key_sright	krIT	%i	KEY_SRIGHT, 0622, sent by shifted right-arrow key	
key_sresume	kRES	%j	KEY_SRSUME, 0623, sent by shifted resume key	
key_ssave	kSAV	!1	KEY_SSAVE, 0624, sent by shifted save key	
key_ssuspend	kSPD	!2	KEY_SSUSPEND, 0625, sent by shifted suspend key	
key_stab	khts	kT	KEY_STAB, 0524, sent by set-tab key	31
key_sundo	kUND	!3	KEY_SUNDO, 0626, sent by shifted undo key	
key_suspend	kspd	&7	KEY_SUSPEND, 0627, sent by suspend key	
key_undo	kund	&8	KEY_UNDO, 0630, sent by undo key	
key_up	kcuu1	ku	KEY_UP, 0403, sent by terminal up-arrow key	31
keypad_local	rmkx	ke	Out of "keypad-transmit" mode	31
keypad_xmit	smkx	ks	Put terminal in "keypad-transmit" mode	31

Variable	Cap-name	Termcap Code	Description	Page#
lab_f0	lf0	10	Labels on function key f0 if not f0	31
lab_f1	lf1	11	Labels on function key f1 if not f1	31
lab_f2	lf2	12	Labels on function key f2 if not f2	31
lab_f3	lf3	13	Labels on function key f3 if not f3	31
lab_f4	lf4	14	Labels on function key f4 if not f4	31
lab_f5	lf5	15	Labels on function key f5 if not f5	31
lab_f6	lf6	16	Labels on function key f6 if not f6	31
lab_f7	lf7	17	Labels on function key f7 if not f7	31
lab_f8	lf8	18	Labels on function key f8 if not f8	31
lab_f9	lf9	19	Labels on function key f9 if not f9	31
lab_f10	lf10	la	Labels on function key f10 if not f10	31
label_off	rmln	LF	Turn off soft labels	31
label_on	smln	LO	Turn on soft labels	31
meta_off	rmm	mo	Turn off "meta mode"	36
meta_on	smm	mm	Turn on "meta mode" (8th bit)	36
micro_column_address	mhpa	ZY	Like column_address for micro adjustment	41
micro_down	mcudl	<td>Like cursor_down for micro adjustment</td> <td>41</td>	Like cursor_down for micro adjustment	41
micro_left	mcubl	Za	Like cursor_left for micro adjustment	41
micro_right	mcufl	Zb	Like cursor_right for micro adjustment	41
micro_row_address	mvpa	Zc	Like row_address for micro adjustment	41
micro_up	mcuul	Zd	Like cursor_up for micro adjustment	41
newline	nel	nw	Newline (behaves like cr followed by lf)	24
order_of_pins	porder	Ze	Matches software bits to print-head pins	47
orig_colors	oc	oc	Set all color(-pair)s to the original ones	35
orig_pair	op	op	Set default color-pair to the original one	35
pad_char	pad	pc	Pad character (rather than null)	35
parm_dch	dch	DC	Delete #1 chars	28
parm_delete_line	dl	DL	Delete #1 lines	26
parm_down_cursor	cud	DO	Move down #1 lines.	26
parm_down_micro	mcud	Zf	Like parm_down_cursor for micro adjust.	41
parm_ich	ich	IC	Insert #1 blank chars	
parm_index	indn	SF	Scroll forward #1 lines.	24
parm_insert_line	il	AL	Add #1 new blank lines	26
parm_left_cursor	cub	LE	Move cursor left #1 spaces	26
parm_left_micro	mcub	Zg	Like parm_left_cursor for micro adjust.	41
parm_right_cursor	cuf	RI	Move right #1 spaces.	26
parm_right_micro	mcuf	Zh	Like parm_right_cursor for micro adjust.	41
parm_rindex	rln	SR	Scroll backward #1 lines.	24
parm_up_cursor	cuu	UP	Move cursor up #1 lines.	26
parm_up_micro	mcuu	Zi	Like parm_up_cursor for micro adjust.	41
pkey_key	pfkey	pk	Prog funct key #1 to type string #2	31
pkey_local	pfloc	pl	Prog funct key #1 to execute string #2	31
pkey_xmit	pfx	px	Prog funct key #1 to xmit string #2	31
plab_norm	pln	pn	Prog label #1 to show string #2	31
print_screen	mc0	ps	Print contents of the screen	36

Variable	Cap-name	Termcap Code	Description	Page#
prtr_non	mc5p	pO	Turn on the printer for #1 bytes	36
prtr_off	mc4	pf	Turn off the printer	36
prtr_on	mc5	po	Turn on the printer	36
repeat_char	rep	rp	Repeat char #1 #2 times	35
req_for_input	rfi	RF	Send next input char (for ptys)	36
reset_1string	rs1	r1	Reset terminal completely to sane modes	32
reset_2string	rs2	r2	Reset terminal completely to sane modes	32
reset_3string	rs3	r3	Reset terminal completely to sane modes	32
reset_file	rf	rf	Name of file containing reset string	32
restore_cursor	rc	rc	Restore cursor to position of last sc	27
row_address	vpa	cv	Vertical position absolute	26
save_cursor	sc	sc	Save cursor position	27
scroll_forward	ind	sf	Scroll text up	23
scroll_reverse	ri	sr	Scroll text down	23
select_char_set	scs	Zj	Select character set	45
set_attributes	sgr	sa	Define the video attributes #1-#9	37
set_background	setb	Sb	Set current background color	34
set_bottom_margin	smgb	Zk	Set bottom margin at current line	43
set_bottom_margin_parm	smgbp	Zl	Set bottom margin at line #1 or #2 lines from bottom	43
set_color_pair	scp	sp	Set current color-pair	35
set_foreground	setf	Sf	Set current foreground color1	34
set_left_margin	smgl	ML	Set left margin at current line	32
set_left_margin_parm	smglp	Zm	Set left (right) margin at column #1 (#2)	43
set_right_margin	smgr	MR	Set right margin at current column	32
set_right_margin_parm	smgrp	Zn	Set right margin at column #1	43
set_tab	hts	st	Set a tab in all rows, current column	31
set_top_margin	smgt	Zo	Set top margin at current line	43
set_top_margin_parm	smgtp	Zp	Set top (bottom) margin at line #1 (#2)	43
set_window	wind	wi	Current window is lines #1-#2 cols #3-#4	
start_bit_image	sbim	Zq	Start printing bit image graphics	47
start_char_set_def	scsd	Zr	Start definition of a character set	45
stop_bit_image	rbim	Zs	End printing bit image graphics	47
stop_char_set_def	rcsd	Zt	End definition of a character set	45
subscript_characters	subcs	Zu	List of "subscript-able" characters	44
superscript_characters	supcs	Zv	List of "superscript-able" characters	44
tab	ht	ta	Tab to next 8-space hardware tab stop	31
these_cause_cr	docr	Zw	Printing any of these chars causes cr	43
to_status_line	tsl	ts	Go to status line, col #1	32
underline_char	uc	uc	Underscore one char and move past it	
up_half_line	hu	hu	Half-line up (reverse 1/2 linefeed)	35
xoff_character	xoffc	XF	X-off character	36
xon_character	xonc	XN	X-on character	36
zero_motion	zerom	Zx	No motion for the subsequent character	43

Booleans

Cap-name	Variable	Termcap Code	Description	Page #
am	auto_right_margin	am	Terminal has automatic margins	24
bw	auto_left_margin	bw	cu l wraps from column 0 to last column	23
ccc	can_change	cc	Terminal can re-define existing color	34
chts	hard_cursor	HC	Cursor is hard to see	29
cpix	cpixel_changes_res	YF	Changing character pitch changes resolution	39
crxm	cr_cancels_micro_modem	YB	Using cr turns off micro mode	42
da	memory_above	da	Display may be retained above the screen	27
daisy	has_print_wheel	YC	Printer needs operator to change character set	45
db	memory_below	db	Display may be retained below the screen	27
eo	erase_overstrike	eo	Can erase overstrikes with a blank	29
eslok	status_line_esc_ok	es	Escape can be used on the status line	32
gn	generic_type	gn	Generic line type (e.g., dialup, switch)	36
hc	hard_copy	hc	Hardcopy terminal	23
hls	hue_lightness_saturation	hl	Terminal uses only HLS color notation (Tektronix)	34
hs	has_status_line	hs	Has extra "status line"	32
hz	tilde_glitch	hz	Hazeltine; can't print tilde (~)	37
in	insert_null_glitch	in	Insert mode distinguishes nulls	27
km	has_meta_key	km	Has a meta key (shift, sets parity bit)	36
lpix	lpixel_changes_res	YG	Changing line pitch changes resolution	39
mc5i	prtr_silent	5i	Printer won't echo on screen	36
mir	move_insert_mode	mi	Safe to move while in insert mode	28
msgr	move_standout_mode	ms	Safe to move in standout modes	29
npc	no_pad_char	NP	Pad character doesn't exist	35
nrrmc	non_rev_rmcup	NR	smcup does not reverse rmcup	26
nxon	needs_xon_xoff	nx	Padding won't work, xon/xoff required	
os	over_strike	os	Terminal overstrikes on hard-copy terminal	23
sam	semi_auto_right_margin	YE	Printing in last column causes cr	42
ul	transparent_underline	ul	Underline character overstrikes	29
xenl	eat_newline_glitch	xn	Newline ignored after 80 columns (<i>Concept</i>)	37
xhp	ceol_standout_glitch	xs	Standout not erased by overwriting (hp)	37
xhpa	col_addr_glitch	YA	Only positive motion for hpa/mhpa caps	41
xon	xon_xoff	xo	Terminal uses xon/xoff handshaking	23
xsb	no_esc_ctlc	xb	Beehive (f1=escape, f2=ctrl C)	37
xt	dest_tabs_magic_smo	xt	Destructive tabs, magic sms o char (t1061)	37
xvpa	row_addr_glitch	YD	Only positive motion for vpa/mvpa caps	41

Numbers

Cap-name	Variable	Termcap Code	Description	Page#
bufsz	buffer_capacity	Ya	Number of bytes buffered before printing	50
colors	max_colors	Co	Maximum number of colors on the screen	34
cols	columns	co	Number of columns in a line	23
cps	print_rate	Ym	Average print rate in characters per second	50
it	init_tabs	it	Tabs initially every # spaces	31
lh	label_height	lh	Number of rows in each label	31
lines	lines	li	Number of lines on a screen or a page	23
lm	lines_of_memory	lm	Lines of memory if > lines; 0 means varies	36
lw	label_width	lw	Number of columns in each label	31
maddr	max_micro_address	Yd	Maximum value in micro_..._address	41
mcs	micro_col_size	Yf	Character step size when in micro mode	39
mjump	max_micro_jump	Ye	Maximum value in parm_..._micro	41
mls	micro_line_size	Yg	Line step size when in micro mode	39
ncv	no_color_video	NC	Video attributes that can't be used with colors	35
nlab	num_labels	Nl	Number of labels on screen (start at 1)	31
npins	number_of_pins	Yh	Number of pins in print-head	47
orc	output_res_char	Yi	Horizontal resolution in units per character	38
orhi	output_res_horz_inch	Yk	Horizontal resolution in units per inch	38
orl	output_res_line	Yj	Vertical resolution in units per line	38
orvi	output_res_vert_inch	Yl	Vertical resolution in units per inch	38
pairs	max_pairs	pa	Maximum number of color-pairs on the screen	34
pb	padding_baud_rate	pb	Lowest baud rate where padding needed	32
spinh	dot_horz_spacing	Yc	Spacing of dots horizontally in dots per inch	47
spinv	dot_vert_spacing	Yb	Spacing of pins vertically in pins per inch	47
vt	virtual_terminal	vt	Virtual terminal number (UNIX system)	
widcs	wide_char_size	Yn	Character step size when in double wide mode	39
wsl	width_status_line	ws	Number of columns in status line	33
xmc	magic_cookie_glitch	sg	Number of blank characters left by smso or rmso	29

Strings

Cap-name	Variable	Termcap Code	Description	Page#
acsc	acs_chars	ac	Graphic charset pairs aAbBcC - def=vt100	33
bel	bell	bl	Audible signal (bell)	23
blink	enter_blink_mode	mb	Turn on blinking	28
bold	enter_bold_mode	md	Turn on bold (extra bright) mode	28
cbt	back_tab	bt	Back tab	31
chr	change_res_horz	ZC	Change horizontal resolution	39
civis	cursor_invisible	vi	Make cursor invisible	29
clear	clear_screen	cl	Clear screen and home cursor	23

Cap-name	Variable	Termcap Code	Description	Page#
cmdch	command_character	CC	Terminal settable cmd character in prototype	36
cnorm	cursor_normal	ve	Make cursor appear normal (undo vs/vi)	29
cpi	change_char_pitch	ZA	Change number of characters per inch	39
cr	carriage_return	cr	Carriage return	23
csnm	char_set_names	Zy	List of character set names	45
csr	change_scroll_region	cs	Change to lines #1 through #2 (vt100)	26
cub	parm_left_cursor	LE	Move cursor left #1 spaces	26
cubl	cursor_left	le	Move left one space.	23
cud	parm_down_cursor	DO	Move down #1 lines.	26
cuf	parm_right_cursor	RI	Move right #1 spaces.	26
cuf1	cursor_right	nd	Non-destructive space (cursor or carriage right)	23
cup	cursor_address	cm	Move to row #1 col #2	24
cuu	parm_up_cursor	UP	Move cursor up #1 lines.	26
cvr	change_res_vert	ZD	Change vertical resolution	39
cvvis	cursor_visible	vs	Make cursor very visible	29
dch	parm_dch	DC	Delete #1 chars	28
dch1	delete_character	dc	Delete character	28
defc	define_char	ZE	Define a character in a character set	45
dim	enter_dim_mode	mh	Turn on half-bright mode	28
dl	delete_line	dl1	Delete line	26
dl	parm_delete_line	DL	Delete #1 lines	26
do	cursor_down	do	Down one line	23
docr	these_cause_cr	Zw	Printing any of these chars causes cr	43
ds1	dis_status_line	ds	Disable status line	32
ech	erase_chars	ec	Erase #1 characters	28
ed	clr_eos	cd	Clear to end of display	26
el	clr_eol	ce	Clear to end of line	26
el1	clr_bol	cb	Clear to beginning of line, inclusive	23
enacs	ena_acs	eA	Enable alternate character set	28
ff	form_feed	ff	Hardcopy terminal page eject	32
flash	flash_screen	vb	Visible bell (may not move cursor)	29
fsl	from_status_line	fs	Return from status line	32
hd	down_half_line	hd	Half-line down (forward 1/2 linefeed)	35
home	cursor_home	ho	Home cursor (if no cup)	26
hpa	column_address	ch	Horizontal position absolute	26
ht	tab	ta	Tab to next 8-space hardware tab stop	31
hts	set_tab	st	Set a tab in all rows, current column	31
hu	up_half_line	hu	Half-line up (reverse 1/2 linefeed)	35
ich	parm_ich	IC	Insert #1 blank chars	
ich1	insert_character	ic	Insert character	28
if	init_file	if	Name of initialization file	24
il	parm_insert_line	AL	Add #1 new blank lines	26
il1	irtinsert_line	al	Add new blank line	26
ind	scroll_forward	sf	Scroll text up	23
indn	parm_index	SF	Scroll forward #1 lines.	24

Cap-name	Variable	Termcap Code	Description	Page#
initc	initialize_color	Ic	Initialize the definition of color	34
initp	initialize_pair	Ip	Initialize color-pair	34
invis	enter_secure_mode	mk	Turn on blank mode (characters invisible)	
ip	insert_padding	ip	Insert pad after character inserted	28
iprogr	init_prog	iP	Path name of program for initialization	31
is1	init_1string	i1	Terminal or printer initialization string	31
is2	init_2string	is	Terminal or printer initialization string	31
is3	init_3string	i3	Terminal or printer initialization string	31
kBEG	key_sbeg	&9	KEY_SBEG, 0572, sent by shifted beginning key	
kCAN	key_scancel	&0	KEY_SCANCEL, 0573, sent by shifted cancel key	
kCMD	key_scommand	*1	KEY_SCOMMAND, 0574, sent by shifted command key	
kCPY	key_scopy	*2	KEY_SCOPY, 0575, sent by shifted copy key	
kCRT	key_screate	*3	KEY_SCREATE, 0576, sent by shifted create key	
kDC	key_sdc	*4	KEY_SDC, 0577, sent by shifted delete-char key	
kDL	key_sdl	*5	KEY_SDL, 0600, sent by shifted delete-line key	
kEND	key_send	*7	KEY_SEND, 0602, sent by shifted end key	
kEOL	key_seol	*8	KEY_SEOL, 0603, sent by shifted clear-line key	
kEXT	key_sexit	*9	KEY_SEXIT, 0604, sent by shifted exit key	
kFND	key_sfind	*0	KEY_SFIND, 0605, sent by shifted find key	
kHLP	key_shelp	#1	KEY_SHELP, 0606, sent by shifted help key	
kHOM	key_shome	#2	KEY_SHOME, 0607, sent by shifted home key	
kIC	key_sic	#3	KEY_SIC, 0610, sent by shifted input key	
kLFT	key_sleft	#4	KEY_SLEFT, 0611, sent by shifted left-arrow key	
kMOV	key_smove	%b	KEY_SMOVE, 0613, sent by shifted move key	
kMSG	key_smessage	%a	KEY_SMESSAGE, 0612, sent by shifted message key	
kNXT	key_snext	%c	KEY_SNEXT, 0614, sent by shifted next key	
kOPT	key_soptions	%d	KEY_SOPTIONS, 0615, sent by shifted options key	
kPRT	key_sprint	%f	KEY_SPRINT, 0617, sent by shifted print key	
kPRV	key_sprevious	%e	KEY_SPREVIOUS, 0616, sent by shifted prev key	
kRDO	key_sredo	%g	KEY_SREDO, 0620, sent by shifted redo key	
kRES	key_sresume	%j	KEY_SRSUME, 0623, sent by shifted resume key	
kRIT	key_sright	%i	KEY_SRIGHT, 0622, sent by shifted right-arrow key	
kRPL	key_sreplace	%h	KEY_SREPLACE, 0621, sent by shifted replace key	
kSAV	key_ssave	!1	KEY_SSAVE, 0624, sent by shifted save key	
kSPD	key_ssuspend	!2	KEY_SSUSPEND, 0625, sent by shifted suspend key	
kUNDO	key_sundo	!3	KEY_SUNDO, 0626, sent by shifted undo key	
ka1	key_a1	K1	KEY_A1, 0534, upper left of keypad	31
ka3	key_a3	K3	KEY_A3, 0535, upper right of keypad	31
kb2	key_b2	K2	KEY_B2, 0536, center of keypad	31
kbeg	key_beg	@1	KEY_BEG, 0542, sent by beg(inning) key	
kbs	key_backspace	kb	KEY_BACKSPACE, 0407, sent by backspace key	31
kc1	key_c1	K4	KEY_C1, 0537, lower left of keypad	31
kc3	key_c3	K5	KEY_C3, 0540, lower right of keypad	31
kcan	key_cancel	@2	KEY_CANCEL, 0543, sent by cancel key	
kcbt	key_btab	kB	KEY_BTAB, 0541, sent by back-tab key	

Cap-name	Variable	Termcap Code	Description	Page#
kclo	key_close	@3	KEY_CLOSE, 0544, sent by close key	
kclr	key_clear	kC	KEY_CLEAR, 0515, sent by clear-screen or erase key	31
kcmd	key_command	@4	KEY_COMMAND, 0545, sent by cmd (command) key	
kcpy	key_copy	@5	KEY_COPY, 0546, sent by copy key	
kcrt	key_create	@6	KEY_CREATE, 0547, sent by create key	
kctab	key_ctab	kt	KEY_CTAB, 0525, sent by clear-tab key	31
kcub1	key_left	kl	KEY_LEFT, 0404, sent by terminal left-arrow key	31
kcud1	key_down	kd	KEY_DOWN, 0402, sent by terminal down-arrow key	31
kcuf1	key_right	kr	KEY_RIGHT, 0405, sent by terminal right-arrow key	31
kcuu1	key_up	ku	KEY_UP, 0403, sent by terminal up-arrow key	31
kdch1	key_dc	kD	KEY_DC, 0512, sent by delete-character key	31
kdll	key_dl	kL	KEY_DL, 0510, sent by delete-line key	31
ked	key_eos	ked	KEY_EOS, 0516, sent by clear-to-end-of-screen key	31
kel	key_eol	kE	KEY_EOL, 0517, sent by clear-to-end-of-line key	31
kend	key_end	@7	KEY_END, 0550, sent by end kee	
kent	key_enter	@8	KEY_ENTER, 0527, sent by enter/send key	
kext	key_exit	@9	KEY_EXIT, 0551, sent by exit key	
kf0	key_f0	k0	KEY_F(0), 0410, sent by function key f0	31
kf1	key_f1	k1	KEY_F(1), 0411, sent by function key f1	31
kf10	key_f10	k;	KEY_F(10), 0422, sent by function key f10	31
kf11	key_f11	F1	KEY_F(11), 0423, sent by function key f11	31
kf12	key_f12	F2	KEY_F(12), 0424, sent by function key f12	31
kf13	key_f13	F3	KEY_F(13), 0425, sent by function key f13	31
kf14	key_f14	F4	KEY_F(14), 0426, sent by function key f14	31
kf15	key_f15	F5	KEY_F(15), 0427, sent by function key f15	31
kf16	key_f16	F6	KEY_F(16), 0430, sent by function key f16	31
kf17	key_f17	F7	KEY_F(17), 0431, sent by function key f17	31
kf18	key_f18	F8	KEY_F(18), 0432, sent by function key f18	31
kf19	key_f19	F9	KEY_F(19), 0433, sent by function key f19	31
kf2	key_f2	k2	KEY_F(2), 0412, sent by function key f2	31
kf20	key_f20	FA	KEY_F(20), 0434, sent by function key f20	31
kf21	key_f21	FB	KEY_F(21), 0435, sent by function key f21	31
kf22	key_f22	FC	KEY_F(22), 0436, sent by function key f22	31
kf23	key_f23	FD	KEY_F(23), 0437, sent by function key f23	31
kf24	key_f24	FE	KEY_F(24), 0440, sent by function key f24	31
kf25	key_f25	FF	KEY_F(25), 0441, sent by function key f25	31
kf26	key_f26	FG	KEY_F(26), 0442, sent by function key f26	31
kf27	key_f27	FH	KEY_F(27), 0443, sent by function key f27	31
kf28	key_f28	FI	KEY_F(28), 0444, sent by function key f28	31
kf29	key_f29	FJ	KEY_F(29), 0445, sent by function key f29	31
kf3	key_f3	k3	KEY_F(3), 0413, sent by function key f3	31
kf30	key_f30	FK	KEY_F(30), 0446, sent by function key f30	31
kf31	key_f31	FL	KEY_F(31), 0447, sent by function key f31	31
kf32	key_f32	FM	KEY_F(32), 0450, sent by function key f32	31
kf33	key_f33	FN	KEY_F(13), 0451, sent by function key f13	31

Cap-name	Variable	Termcap Code	Description	Page#
kf34	key_f34	FO	KEY_F(34), 0452, sent by function key f34	31
kf35	key_f35	FP	KEY_F(35), 0453, sent by function key f35	31
kf36	key_f36	FQ	KEY_F(36), 0454, sent by function key f36	31
kf37	key_f37	FR	KEY_F(37), 0455, sent by function key f37	31
kf38	key_f38	FS	KEY_F(38), 0456, sent by function key f38	31
kf39	key_f39	FT	KEY_F(39), 0457, sent by function key f39	31
kf4	key_f4	k4	KEY_F(4), 0414, sent by function key f4	31
kf40	key_f40	FU	KEY_F(40), 0460, sent by function key f40	31
kf41	key_f41	FV	KEY_F(41), 0461, sent by function key f41	31
kf42	key_f42	FW	KEY_F(42), 0462, sent by function key f42	31
kf43	key_f43	FX	KEY_F(43), 0463, sent by function key f43	31
kf44	key_f44	FY	KEY_F(44), 0464, sent by function key f44	31
kf45	key_f45	FZ	KEY_F(45), 0465, sent by function key f45	31
kf46	key_f46	Fa	KEY_F(46), 0466, sent by function key f46	31
kf47	key_f47	Fb	KEY_F(47), 0467, sent by function key f47	31
kf48	key_f48	Fc	KEY_F(48), 0470, sent by function key f48	31
kf49	key_f49	Fd	KEY_F(49), 0471, sent by function key f49	31
kf5	key_f5	k5	KEY_F(5), 0415, sent by function key f5	31
kf50	key_f50	Fe	KEY_F(50), 0472, sent by function key f50	31
kf51	key_f51	Ff	KEY_F(51), 0473, sent by function key f51	31
kf52	key_f52	Fg	KEY_F(52), 0474, sent by function key f52	31
kf53	key_f53	Fh	KEY_F(53), 0475, sent by function key f53	31
kf54	key_f54	Fi	KEY_F(54), 0476, sent by function key f54	31
kf55	key_f55	Fj	KEY_F(55), 0477, sent by function key f55	31
kf56	key_f56	Fk	KEY_F(56), 0500, sent by function key f56	31
kf57	key_f57	Fl	KEY_F(57), 0501, sent by function key f57	31
kf58	key_f58	Fm	KEY_F(58), 0502, sent by function key f58	31
kf59	key_f59	Fn	KEY_F(59), 0503, sent by function key f59	31
kf6	key_f6	k6	KEY_F(6), 0416, sent by function key f6	31
kf60	key_f60	Fo	KEY_F(60), 0504, sent by function key f60	31
kf61	key_f61	Fp	KEY_F(61), 0505, sent by function key f61	31
kf62	key_f62	Fq	KEY_F(62), 0506, sent by function key f62	31
kf63	key_f63	Fr	KEY_F(63), 0507, sent by function key f63	31
kf7	key_f7	k7	KEY_F(7), 0417, sent by function key f7	31
kf8	key_f8	k8	KEY_F(8), 0420, sent by function key f8	31
kf9	key_f9	k9	KEY_F(9), 0421, sent by function key f9	31
kfnd	key_find	@0	KEY_FIND, 0552, sent by find key	
khlp	key_help	%1	KEY_HELP, 0553, sent by help key	
khome	key_home	kh	KEY_HOME, 0406, sent by home key	31
khts	key_stab	kT	KEY_STAB, 0524, sent by set-tab key	31
kich1	key_ic	kI	KEY_IC, 0513, sent by ins-char/enter ins-mode key	31
kill	key_il	kA	KEY_IL, 0511, sent by insert-line key	31
kind	key_sf	kF	KEY_SF, 0520, sent by scroll-forward/down key	31
kll	key_ll	kH	KEY_LL, 0533, sent by home-down key	31
kmove	key_move	%4	KEY_MOVE, 0556, sent by move key	

Cap-name	Variable	Termcap Code	Description	Page#
kmrk	key_mark	%2	KEY_MARK, 0554, sent by mark key	
kmsg	key_message	%3	KEY_MESSAGE, 0555, sent by message key	
knp	key_npage	kN	KEY_NPAGE, 0522, sent by next-page key	31
knxt	key_next	%5	KEY_NEXT, 0557, sent by next-object key	
kopn	key_open	%6	KEY_OPEN, 0560, sent by open key	
kopt	key_options	%7	KEY_OPTIONS, 0561, sent by options key	
kpp	key_ppage	kP	KEY_PPAGE, 0523, sent by previous-page key	31
kprt	key_print	%9	KEY_PRINT, 0532, sent by print or copy key	
kprv	key_previous	%8	KEY_PREVIOUS, 0562, sent by previous-object key	
krdo	key_redo	%0	KEY_REDO, 0563, sent by redo key	
kref	key_reference	&1	KEY_REFERENCE, 0564, sent by ref(erence) key	
kres	key_resume	&5	KEY_RESUME, 0570, sent by resume key	
krfr	key_refresh	&2	KEY_REFRESH, 0565, sent by refresh key	
kri	key_sr	kR	KEY_SR, 0521, sent by scroll-backward/up key	31
krmir	key_eic	kM	KEY_EIC, 0514, sent by rmir or smir in insert mode	31
krpl	key_replace	&3	KEY_REPLACE, 0566, sent by replace key	
krst	key_restart	&4	KEY_RESTART, 0567, sent by restart key	
ksav	key_save	&6	KEY_SAVE, 0571, sent by save key	
kslt	key_select	*6	KEY_SELECT, 0601, sent by select key	
kspd	key_suspend	&7	KEY_SUSPEND, 0627, sent by suspend key	
ktbc	key_catab	ka	KEY_CATAB, 0526, sent by clear-all-tabs key	31
kund	key_undo	&8	KEY_UNDO, 0630, sent by undo key	
lf0	lab_f0	l0	Labels on function key f0 if not f0	31
lf1	lab_f1	l1	Labels on function key f1 if not f1	31
lf10	lab_f10	lA	Labels on function key f10 if not f10	31
lf2	lab_f2	l2	Labels on function key f2 if not f2	31
lf3	lab_f3	l3	Labels on function key f3 if not f3	31
lf4	lab_f4	l4	Labels on function key f4 if not f4	31
lf5	lab_f5	l5	Labels on function key f5 if not f5	31
lf6	lab_f6	l6	Labels on function key f6 if not f6	31
lf7	lab_f7	l7	Labels on function key f7 if not f7	31
lf8	lab_f8	l8	Labels on function key f8 if not f8	31
lf9	lab_f9	l9	Labels on function key f9 if not f9	31
ll	cursor_to_ll	ll	Last line, first column (if no cup)	26
lpi	change_line_pitch	ZB	Change number of lines per inch	39
mc0	print_screen	ps	Print contents of the screen	36
mc4	prtr_off	pf	Turn off the printer	36
mc5	prtr_on	po	Turn on the printer	36
mc5p	prtr_non	pO	Turn on the printer for #1 bytes	36
mcub	parm_left_micro	Zg	Like parm_left_cursor for micro adjust.	41
mcub1	micro_left	Za	Like cursor_left for micro adjustment	41
mcud	parm_down_micro	Zf	Like parm_down_cursor for micro adjust.	41
mcud1	micro_down	ZZ	Like cursor_down for micro adjustment	41
mcuf	parm_right_micro	Zh	Like parm_right_cursor for micro adjust.	41
mcuf1	micro_right	Zb	Like cursor_right for micro adjustment	41

Cap-name	Variable	Termcap Code	Description	Page#
mcuu	parm_up_micro	Zi	Like <code>parm_up_cursor</code> for micro adjust.	41
mcuul	micro_up	Zd	Like <code>cursor_up</code> for micro adjustment)	41
mgc	clear_margins	MC	Clear all margins (top, bottom, and sides)	32
mhpa	micro_column_address	ZY	Like <code>column_address</code> for micro adjustment	41
mrcup	cursor_mem_address	CM	Memory relative cursor addressing	24
mvpa	micro_row_address	Zc	Like <code>row_address</code> for micro adjustment	41
nel	newline	nw	Newline (behaves like <code>cr</code> followed by <code>lf</code>)	24
oc	orig_colors	oc	Set all color(-pair)s to the original ones	35
op	orig_pair	op	Set default color-pair to the original one	35
pad	pad_char	pc	Pad character (rather than null)	35
pfkey	pkey_key	pk	Prog funct key #1 to type string #2	31
pfloc	pkey_local	pl	Prog funct key #1 to execute string #2	31
pfx	pkey_xmit	px	Prog funct key #1 to xmit string #2	31
pln	plab_norm	pn	Prog label #1 to show string #2	31
porder	order_of_pins	Ze	Matches software bits to print-head pins	47
prot	enter_protected_mode	mp	Turn on protected mode	28
rbim	stop_bit_image	Zs	End printing bit image graphics	47
rc	restore_cursor	rc	Restore cursor to position of last <code>sc</code>	27
rcsd	stop_char_set_def	Zt	End definition of a character set	45
rep	repeat_char	rp	Repeat char #1 #2 times	35
rev	enter_reverse_mode	mr	Turn on reverse video mode	28
rf	reset_file	rf	Name of file containing reset string	32
rfi	req_for_input	RF	Send next input char (for <code>pty</code> s)	36
ri	scroll_reverse	sr	Scroll text down	23
rin	parm_rindex	SR	Scroll backward #1 lines.	24
ritm	exit_italics_mode	ZR	Disable italics	44
rlm	exit_leftward_mode	ZS	Enable rightward (normal) carriage motion	42
rmacs	exit_alt_charset_mode	ae	End alternate character set	
rmam	exit_am_mode	RA	Turn off automatic margins	
rmcup	exit_ca_mode	te	String to end programs that use <code>cup</code>	26
rmdc	exit_delete_mode	ed	End delete mode	28
rmicm	exit_micro_mode	ZT	Disable micro motion capabilities	42
rmir	exit_insert_mode	ei	End insert mode	28
rmkx	keypad_local	ke	Out of "keypad-transmit" modey	31
rmln	label_off	LF	Turn off soft labels	31
rmm	meta_off	mo	Turn off "meta mode"	36
rmp	char_padding	rP	Like <code>ip</code> but when in replace mode	28
rmso	exit_standout_mode	se	End standout mode	29
rmul	exit_underline_mode	ue	End underscore mode	29
rmxon	exit_xon_mode	RX	Turn off <code>xon/xoff</code> handshaking	36
rs1	reset_1string	r1	Reset terminal completely to sane modes	32
rs2	reset_2string	r2	Reset terminal completely to sane modes	32
rs3	reset_3string	r3	Reset terminal completely to sane modes	32
rshm	exit_shadow_mode	ZU	Disable shadow printing	44
rsubm	exit_subscript_mode	ZV	Disable subscript printing	44

Cap-name	Variable	Termcap Code	Description	Page #
rsupm	exit_superscript_mode	ZW	Disable superscript printing	44
rum	exit_upward_mode	ZX	Enable downward (normal) carriage motion	42
rwidm	exit_doublewide_mode	ZQ	Disable double wide printing	44
sbim	start_bit_image	Zq	Start printing bit image graphics	47
sc	save_cursor	sc	Save cursor position	27
scp	set_color_pair	sp	Set current color-pair	35
scs	select_char_set	Zj	Select character set	45
scsc	start_char_set_def	Zr	Start definition of a character set	45
sdrfq	enter_draft_quality	ZG	Set draft quality print	49
setb	set_background	Sb	Set current background color	34
setf	set_foreground	Sf	Set current foreground color	34
sgr	set_attributes	sa	Define the video attributes #1-#9	37
sgr0	exit_attribute_mode	me	Turn off all attributes	28
sitm	enter_italics_mode	ZH	Enable italics	44
slm	enter_leftward_mode	ZI	Enable leftward carriage motion	42
smacs	enter_alt_charset_mode	as	Start alternate character set	28
smam	enter_am_mode	SA	Turn on automatic margins	
smcup	enter_ca_mode	ti	String to begin programs that use cup	26
smdc	enter_delete_mode	dm	Delete mode (enter)	28
smgb	set_bottom_margin	Zk	Set bottom margin at current line	43
smgbp	set_bottom_margin_parm	Zl	Set bottom margin at line #1 or #2 lines from bottom	43
smgl	set_left_margin	ML	Set left margin at current line	32
smglp	set_left_margin_parm	Zm	Set left (right) margin at column #1 (#2)	43
smgr	set_right_margin	MR	Set right margin at current column	32
smgrp	set_right_margin_parm	Zn	Set right margin at column #1	43
smgt	set_top_margin	Zo	Set top margin at current line	43
smgtp	set_top_margin_parm	Zp	Set top (bottom) margin at line #1 (#2)	43
smicm	enter_micro_mode	ZJ	Enable micro motion capabilities	42
smir	enter_insert_mode	im	Insert mode (enter)	28
smkx	keypad_xmit	ks	Put terminal in "keypad-transmit" mode	31
smln	label_on	LO	Turn on soft labels	31
smm	meta_on	mm	Turn on "meta mode" (8th bit)	36
smso	enter_standout_mode	so	Begin standout mode	29
smul	enter_underline_mode	us	Start underscore mode	29
smxon	enter_xon_mode	SX	Turn on xon/xoff handshaking	36
snlq	enter_near_letter_quality	ZK	Set near-letter quality print	49
snrmq	enter_normal_quality	ZL	Set normal quality print	49
sshm	enter_shadow_mode	ZM	Enable shadow printing	44
ssubm	enter_subscript_mode	ZN	Enable subscript printing	44
rsupm	enter_superscript_mode	ZO	Enable superscript printing	44
subcs	subscript_characters	Zu	List of "subscript-able" characters	44
sum	enter_upward_mode	ZP	Enable upward carriage motion	42
supcs	superscript_characters	Zv	List of "superscript-able" characters	44
swidm	enter_doublewide_mode	ZF	Enable double wide printing	44
tbc	clear_all_tabs	ct	Clear all tab stops	31

Types of Capabilities in the Sample Entry

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 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, enclosed in \$<..> brackets, as in *el*=\EK\$<3>. Padding characters are supplied by *tputs()*. The delay can be any of the following: a number (5), a number followed by a '*' (5*), a number followed by a '/' (5/), or a number followed by both (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 (e.g., \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.

Preparing Descriptions

The most effective way to prepare a device description is by imitating the description of a similar device in *terminfo* and building up a description gradually, using partial descriptions with *vi(1)* 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(1)* to work with that device. To test a new device description, set the environment variable **TERMINFO** to the pathname of a directory containing the compiled description you are working on and programs will look there rather than in */usr/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(1)*, delete 16 or so lines from the middle of the screen, and then hit 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.

Section 1-1: Basic Capabilities

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 **cr**. (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 **cub1**. Similarly, sequences to move to the right, up, and down should be given as **cuf1**, **cuu1**, and **cud1**, respectively. These local cursor motions must not alter the text they pass over; for example, you would not normally use "**cuf1=\s**" because the space would erase the character moved over.

A VERY IMPORTANT POINT HERE IS THAT THE LOCAL CURSOR MOTIONS ENCODED IN *terminfo* ARE UNDEFINED AT THE LEFT AND TOP EDGES OF A SCREEN TERMINAL. PROGRAMS SHOULD NEVER ATTEMPT TO BACKSPACE AROUND THE LEFT EDGE, UNLESS **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 **ind** (INDEX) STRING.

TO SCROLL TEXT DOWN, A PROGRAM GOES TO THE TOP LEFT CORNER OF THE SCREEN AND SENDS THE **ri** (REVERSE INDEX) STRING. THE STRINGS **ind** AND **ri** ARE UNDEFINED WHEN NOT ON THEIR RESPECTIVE CORNERS OF THE SCREEN.

Parameterized versions of the scrolling sequences are **indn** and **rin**. These versions have the same semantics as **ind** and **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 **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 **cuf1** from the last column. Backward motion from the left edge of the screen is possible only when **bw** is specified. In this case, **cub1** will move to the right edge of the previous row. If **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, **am** should be specified in the *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 **nel** (newline). It does not matter if the command clears the remainder of the current line, so if the device has no **cr** and **lf** it may still be possible to craft a working **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:

```
5320|att5320|AT&T 5320 hardcopy terminal,
    am, hc, os,
    cols#132,
    bel=^G, cr=^r, cub1=^b, cnd1=^n,
    dch1=^E[P, dl1=^E[M,
    ind=^n,
```

while the Lear Siegler ADM-3 is described as

```
adm3|lsi adm3,
    am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
    cud1=^J, ind=^J, lines#24,
```

Section 1-2: Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with **printf(3S)**-like escapes (**%x**) in it. For example, to address the cursor, the **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
 print pop() gives %c

%c

%p[1-9] push *i*th parm
 %P[a-z] set variable [a-z] to pop()
 %g[a-z] get 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 *integer*₂() op pop *integer*₁())
 %& %| %^ bit operations: push(pop *integer*₂() op pop *integer*_{sub 1}())
 %= %> %< logical operations: push(pop() op pop())
 %A %O logical operations: and, or
 %! %~ unary operations: push(op pop())
 %i (for ANSI terminals)
 add 1 to first parm, if one parm present,
 or first two parms, if more than one parm present

%? expr %t thenpart %e elsepart %;
 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 %;
*c*_i are conditions, *b*_i are bodies.

If the "-" flag is used with "[%doxXs]", then a colon (:) must be placed between the "%" and the "-" to differentiate the flag from the binary "%-" operator, e.g "%:-16.16s".

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent `\E&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=\E&a%p2%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 (`cu1`), and to move the cursor up one line on the screen (`cuu`). 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.

Section 1-3: Cursor Motions

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 **ll**; this may involve going up with **cuu1** from the home position, but a program should never do this itself (unless **ll** 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 (e.g., move *n* spaces to the right) these can be given as **cud**, **cub**, **cuf**, and **cuu** with a single parameter indicating how many spaces to move. These are primarily useful if the device does not have **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 **smcup** and **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 **smcup** sets the command character to be the one used by **terminfo**. If the **smcup** sequence will not restore the screen after an **rmcup** sequence is output (to the state prior to outputting **rmcup**), specify **nrrmc**.

Section 1-4: Area Clears

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 **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 **el1**. If the terminal can clear from the current position to the end of the display, then this should be given as **ed**. **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 **ed** is not available.)

Section 1-5: Insert/Delete Line

If the terminal can open a new blank line before the line where the cursor is, this should be given as **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 **dl1**; this is done only from the first position on the line to be deleted. Versions of **il1** and **dl1** which take a single parameter and insert or delete that many lines can be given as **il** and **dl**.

If the terminal has a settable destructive scrolling region (like the VT100) the command to set this can be described with the **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 `sc` and `rc` (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using `ri` or `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 (`ri`) followed by a delete line (`dl1`) or index (`ind`). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the `dl1` or `ind`, then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify `csr` if the terminal has non-destructive scrolling regions, unless `ind`, `ri`, `indn`, `rin`, `dl`, 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.

Section 1-6: Insert/Delete Character

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 (e.g., if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability **mir** to speed up inserting in this case. Omitting **mir** will affect only speed. Some terminals (notably Datamedia's) must not have **mir** because of the way their insert mode works.

Finally, you can specify **dch1** to delete a single character, **dch** with one parameter, *n*, to delete *n* characters, and delete mode by giving **smdc** and **rmdc** to enter and exit delete mode (any mode the terminal needs to be placed in for **dch1** to work).

A command to erase *n* characters (equivalent to outputting *n* blanks without moving the cursor) can be given as **ech** with one parameter.

Section 1-7: Highlighting, Underlining, and Visible Bells

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 (**blink**), bold or extra-bright characters (**bold**), dim or half-bright characters (**dim**), blanking or invisible text (**invis**), protected text (**prot**), a reverse-video screen (**rev**), and an alternate character set (**smacs** to enter this mode and **rmacs** to exit it). (If a command is necessary before you can enter alternate character set mode, give the sequence in **enacs** or "enable alternate-character-set" mode.) Turning on any of these modes singly may or may not turn off other modes.

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 **dim** or **blink**.

You should choose one display method as *standout mode* (see *curses(3X)*) 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 **smso** and **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 **xmc** should be given to tell how many spaces are left.

Sequences to begin underlining and end underlining can be specified as **smul** and **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 **uc**.

Terminals with the "magic cookie" glitch (**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 **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 **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 **cvvis**. The boolean **chts** should also be given. If there is a way to make the cursor completely invisible, give that as **civis**. The capability **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 **ul**. For devices on which a character overstriking another leaves both characters on the screen, specify the capability **os**. If overstrikes are erasable with a blank, then this should be indicated by specifying **eo**.

If there is a sequence to set arbitrary combinations of modes, this should be given as **sg**r (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 **sg**r; 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.

tparam parameter	attribute	escape sequence
	none	\E[0m
p1	standout	\E[0;4;7m
p2	underline	\E[0;3m
p3	reverse	\E[0;4m
p4	blink	\E[0;5m
p5	dim	\E[0;7m
p6	bold	\E[0;3;4m
p7	invis	\E[0;8m
p8	protect	not available
p9	altcharset	^O (off) ^N(on)

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:

sequence	when to output	terminfo translation
\E[0	always	\E[0
;3	if p2 or p6	%%?%p2%p6%?t;3%;
;4	if p1 or p3 or p6	%%?%p1%p3%p6%?t;4%;
;5	if p4	%%?%p4%t;5%;
;7	if p1 or p5	%%?%p1%p5%?t;7%;
;8	if p7	%%?%p7%t;8%;
m	always	m
^N or ^O	if p9 ^N, else ^O	%%?%p9%t^N%e^O%;

Putting this all together into the `sgr` sequence gives:

```
sgr=\E[0%%?%p2%p6%?t;3%;%%?%p1%p3%p6%?t;4%;%%?%p5%t;5%;%%?%p1%p5%
?t;7%;%%?%p7%t;8%;m%%?%p9%t^N%e^O%;
```

REMEMBER THAT `sgr` AND `sgr0` MUST ALWAYS BE SPECIFIED.

Section 1-8: Keypad

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**, **kcuf1**, **kcuu1**, **kcud1**, 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 transmitted by certain other special keys can be given: **kll** (home down), **kbs** (backspace), **ktbc** (clear all tabs), **kctab** (clear the tab stop in this column), **kclr** (clear screen or erase key), **kdch1** (delete character), **kdl1** (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), **knf** (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 3 by 3 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 **pxf**. 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 **pxf** 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.

Section 1-9: Tabs and Initialization

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; **ipro**, 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 **ipro**; 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(1)**; see **profile(4)**.

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 harder 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/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(7)** 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, e.g., **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.

glyph name	vt100+ character
arrow pointing right	+
arrow pointing left	,
arrow pointing down	.
solid square block	0
lantern symbol	I
arrow pointing up	-
diamond	'
checker board (stipple)	a
degree symbol	f
plus/minus	g
board of squares	h
lower right corner	j
upper right corner	k
upper left corner	l
lower left corner	m
plus	n
scan line 1	o
horizontal line	q
scan line 9	s
left tee (├)	t
right tee (─)	u
bottom tee (┴)	v
top tee (┤)	w
vertical line	x
bullet	~

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,

glyph name	vt100+ char	new tty char
upper left corner	l	R
lower left corner	m	F
upper right corner	k	T
lower right corner	j	G
horizontal line	q	
vertical line	x	

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.

Section 1-13: Color Manipulation

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 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 (ranging from 0 to 1000).

Tektronix 4100 series terminals use a type of color notation called HLS (Hue Lightness Saturation) instead of RGB color notation. For such terminals one must define a boolean variable **hls**. The last three arguments to the **initc** string would then be HLS values: *H*, ranging from 0 to 360; and *L* and *S*, ranging from 0 to 100.

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 **setf** (set foreground) and **setb** (set 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.

Attribute	Bit Position	Decimal Value
A_STANDOUT	0	1
A_UNDERLINE	1	2
A_REVERSE	2	4
A_BLINK	3	8
A_DIM	4	16
A_BOLD	5	32
A_INVIS	6	64
A_PROTECT	7	128
A_ALTCHARSET	8	256

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).

Section 1-14: Miscellaneous

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, **tparm(repeat_char, 'x', 10)** is the same as **xxxxxxxxxx**.

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 UNIX 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 UNIX 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 **rft**.

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.

Section 1-15: Special Cases

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 can not 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.

Section 1-16: Similar Terminals

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!Teletype 4424 in display function group
  ii, rev@, sgr@, smul@, use=att4424,
```

defines an AT&T 4424 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.

PART 2: PRINTER CAPABILITIES

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.

Section 2-1: Rounding Values

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.

Section 2-2: Printer Resolution

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 *terminfo* 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. *terminfo* 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 *terminfo* 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 *terminfo* 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.

Section 2-3: Specifying Printer Resolution'

The printing resolution of a printer is given in several ways. Each specifies the resolution as the number of smallest steps per distance:

Specification of Printer Resolution	
<u>Characteristic Number of Smallest Steps</u>	
orhi	Steps per inch horizontally
orvi	Steps per inch vertically
orc	Steps per column
orl	Steps per line

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):

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").

Specification of Printer Resolution
Effects of Changing the Character/Line Pitches

<i>Before</i>	<i>After</i>
<i>Using cpi with cpix clear:</i>	
orhi ′	orhi
orc ′	$\text{orc} = \frac{\text{orhi}}{V_{cpi}}$
<i>Using cpi with cpix set:</i>	
orhi ′	$\text{orhi} = \text{orc} \cdot V_{cpi}$
orc ′	orc
<i>Using lpi with lpix clear:</i>	
orvi ′	orvi
orl ′	$\text{orl} = \frac{\text{orvi}}{V_{lpi}}$
<i>Using lpi with lpix set:</i>	
orvi ′	$\text{orvi} = \text{orl} \cdot V_{lpi}$
orl ′	orl
<i>Using chr:</i>	
orhi ′	orhi
orc ′	V_{chr}
<i>Using cvr:</i>	
orvi ′	orvi
orl ′	V_{cvr}
<i>Using cpi or chr:</i>	
widcs ′	$\text{widcs} = \text{widcs}' \cdot \frac{\text{orc}}{\text{orc}'}$
mcs ′ †	$\text{mcs} = \text{mcs}' \cdot \frac{\text{orc}}{\text{orc}'}$

V_{cpi} , V_{lpi} , V_{chr} , and V_{cvt} are the arguments used with **cpi**, **lpi**, **chr**, and **cvt**, respectively. The † mark indicates the old value.

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.

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

mcub1	Move 1 step left
mcuf1	Move 1 step right
mcuu1	Move 1 step up
mcud1	Move 1 step down
mcub	Move <i>N</i> steps left
mcuf	Move <i>N</i> steps right
mcuu	Move <i>N</i> steps up
mcud	Move <i>N</i> steps down
mhpa	Move <i>N</i> steps from the left
mvpa	Move <i>N</i> 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. *terminfo* has capabilities for specifying these limits.

Limits to Motion

mjump	Limit on use of mcub1 , mcuf1 , mcuu1 , mcud1
maddr	Limit on use of mhpa , mvpa
xhpa	If set, hpa and mhpa can't move left
xvpa	If set, vpa and 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
crxm	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 <i>N</i> steps right
mcuf	Move <i>N</i> steps left
cub1	Move 1 column right
cuf1	Move 1 column left
cub	Move <i>N</i> columns right
cuf	Move <i>N</i> columns left

While sense of vertical motions reversed:

mcuu1	Move 1 step down
mcud1	Move 1 step up
mcuu	Move <i>N</i> steps down
mcud	Move <i>N</i> steps up
cuu1	Move 1 line down
cud1	Move 1 line up
cuu	Move <i>N</i> lines down
cud	Move <i>N</i> lines up

The reverse motion modes should not affect the **mvp**a and **mhp**a 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

docr	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 <i>N</i>
smglp	Set left margin at column <i>N</i>
smgrp	Set right margin at column <i>N</i>
smgtp	Set top margin at line <i>N</i>

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 **smgtp** and **smgbp** are set, each is used to set the top and bottom margin, respectively: **smgtp** 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 *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 **smgpb** 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 **smgtp** and **smgpb** 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 *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 **smgtp** and **smgpb** 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 **smgpb** means the bottom line of the page.

All margins can be cleared with **mgc**.

Shadows, Italics, Wide Characters, Superscripts, Subscripts

Five new sets of strings are used to describe the capabilities printers have of enhancing printed text.

Enhanced Printing	
sshm	Enter shadow-printing mode
rshm	Exit shadow-printing mode
sitm	Enter italicizing mode
ritm	Exit italicizing mode
swidm	Enter wide character mode
rwidm	Exit wide character mode
ssupm	Enter superscript mode
rsupm	Exit superscript mode
supcs	List of characters available as superscripts
ssubm	Enter subscript mode
rsubm	Exit subscript mode
subcs	List of characters available as subscripts

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 **ssubm/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 *ssubm* 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: Bi B_i Bⁱ

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.

Section 2-5: Alternate Character Sets

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	
<i>scs</i>	Select character set <i>N</i>
<i>scsd</i>	Start definition of character set <i>N</i> , <i>M</i> characters
<i>defc</i>	Define character <i>A</i> , <i>B</i> dots wide, descender <i>D</i>
<i>rcsd</i>	End definition of character set <i>N</i>
<i>csnm</i>	List of character set names
<i>daisy</i>	Printer has manually changed print-wheels

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.

Section 2-6: Dot-Matrix Graphics

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 a 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 0 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.

Section 2-7: Effect of Changing Printing Resolution

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	
<i>Before</i>	<i>After</i>
<i>Using cpi with cpix clear:</i>	
spinh '	spinh
<i>Using cpi with cpix set:</i>	
spinh '	spinh=spinh' , $\frac{\text{orhi}}{\text{orhi}'}$
<i>Using lpi with lpix clear:</i>	
spinv '	spinv

Dot-Matrix Graphics	
Effects of Changing the Character/Line Pitches	
Before	After
<i>Using lpi with lpix set:</i>	
spinv '	spinv=spinv' $\frac{\text{orhi}}{\text{orhi}'}$
<i>Using chr:</i>	
spinh '	spinh
<i>Using cvr:</i>	
spinv '	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.

Section 2-8: Print Quality

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
srmq	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.

Section 2-9: Printing Rate and Buffer Size

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

<code>/usr/lib/terminfo/?/*</code>	compiled terminal description database
<code>/usr/lib/.COREterm/?/*</code>	subset of compiled terminal description database
<code>/usr/lib/tabset/*</code>	tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs)

SEE ALSO

`curses(3X)`, `printf(3S)` in the *Programmer's Reference Manual*.
`captainfo(1M)`, `infocmp(1M)`, `tic(1M)`, `term(5)`, `tty(7)` in the *System Administrator's Reference Manual*.
`tput(1)` in the *User's Reference Manual*.
 Chapter 10 of the *Programmer's Guide*.

WARNING

As described in the "Tabs and Initialization" section above, a terminal's initialization strings, *is1*, *is2*, and *is3*, if defined, must be output before a *curses(3X)* program is run. An available mechanism for outputting such strings is *tput init* (see *tput(1)* and *profile(4)*).

If a null character (`\0`) is encountered in a string, the null and all characters after it are lost. Therefore it is not possible to code a null character (`\0`) and send it to a device (either terminal or printer). The suggestion of sending a `\0200`, where a `\0` (null) is needed can succeed only if the device (terminal or printer) ignores the eighth bit. For example, because all eight bits are used in the standard international ASCII character set, devices that adhere to this standard will treat `\0200` differently from `\0`.

Tampering with entries in `/usr/lib/.COREterm/?/*` or `/usr/lib/terminfo/?/*` (for example, changing or removing an entry) can affect programs such as *vi(1)* that expect the entry to be present and correct. In particular, removing the description for the "dumb" terminal will cause unexpected problems.

NOTE

The *termcap* database (from earlier releases of UNIX System V) may not be supplied in future releases.



NAME

timezone – set default system time zone

SYNOPSIS

/etc/TIMEZONE

DESCRIPTION

This file sets and exports the time zone environmental variable TZ.

This file is "dotted" into other files that must know the time zone.

The syntax of TZ can be described as follows:

```

TZ                →                zone
                  | zone signed_time
                  | zone signed_time zone
                  | zone signed_time zone dst
zone              →                letter letter letter
signed_time      →                sign time
                  | time
time             →                hour
                  | hour : minute
                  | hour : minute : second
dst              →                signed_time
                  | signed_time ; dst_date , dst_date
                  | ; dst_date , dst_date
dst_date         →                julian
                  | julian / time
letter           →                a | A | b | B | ... | z | Z
hour             →                00 | 01 | ... | 23
minute          →                00 | 01 | ... | 59
second          →                00 | 01 | ... | 59
julian          →                001 | 002 | ... | 366
sign            →                - | +
  
```

EXAMPLES

The contents of /etc/TIMEZONE corresponding to the simple example below could be

```

#      Time Zone
TZ=EST5EDT
export TZ
  
```

A simple setting for New Jersey could be

```
TZ=EST5EDT
```

where EST is the abbreviation for the main time zone, 5 is the difference, in hours, between GMT (Greenwich Mean Time) and the main time zone, and EDT is the abbreviation for the alternate time zone.

The most complex representation of the same setting, for the year 1986, is

```
TZ="EST5:00:00EDT4:00:00;117/2:00:00,299/2:00:00"
```

where EST is the abbreviation for the main time zone, 5:00:00 is the difference, in hours, minutes, and seconds between GMT and the main time zone, EDT is

the abbreviation for the alternate time zone, 4:00:00 is the difference, in hours, minutes, and seconds between GMT and the alternate time zone, 117 is the number of the day of the year (Julian day) when the alternate time zone will take effect, 2:00:00 is the number of hours, minutes, and seconds past midnight when the alternate time zone will take effect, 299 is the number of the day of the year when the alternate time zone will end, and 2:00:00 is the number of hours, minutes, and seconds past midnight when the alternate time zone will end.

A southern hemisphere setting such as the Cook Islands could be

TZ="KDT9:30KST10:00;64/5:00,303/20:00"

This setting means that KDT is the abbreviation for the main time zone, KST is the abbreviation for the alternate time zone, KST is 9 hours and 30 minutes later than GMT, KDT is 10 hours later than GMT, the starting date of KDT is the 64th day at 5 AM, and the ending date of KDT is the 303rd day at 8 PM.

Starting and ending times are relative to the alternate time zone. If the alternate time zone start and end dates and the time are not provided, the days for the United States that year will be used and the time will be 2 AM. If the start and end dates are provided but the time is not provided, the time will be midnight.

Note that in most installations, TZ is set to the correct value by default when the user logs on, via the local */etc/profile* file (see *profile(4)*).

NOTES

When the longer format is used, the TZ variable must be surrounded by double quotes as shown.

The system administrator must change the Julian start and end days annually if the longer form of the TZ variable is used.

Setting the time during the interval of change from the main time zone to the alternate time zone or vice versa can produce unpredictable results.

SEE ALSO

rc2(1M), profile(4), environ(5).
ctime(3C) in the *Programmer's Reference Manual*.

NAME

unistd – file header for symbolic constants

SYNOPSIS

```
#include <unistd.h>
```

DESCRIPTION

The header file *<unistd.h>* lists the symbolic constants and structures not already defined or declared in some other header file.

```
/* Symbolic constants for the "access" routine: */
```

```
#define R_OK      4      /*Test for Read permission */
#define W_OK      2      /*Test for Write permission */
#define X_OK      1      /*Test for eXecute permission */
#define F_OK      0      /*Test for existence of File */
```

```
#define F_ULOCK  0      /*Unlock a previously locked region */
#define F_LOCK   1      /*Lock a region for exclusive use */
#define F_TLOCK  2      /*Test and lock a region for exclusive use */
#define F_TEST   3      /*Test a region for other processes locks */
```

```
/*Symbolic constants for the "lseek" routine: */
```

```
#define SEEK_SET  0      /* Set file pointer to "offset" */
#define SEEK_CUR  1      /* Set file pointer to current plus "offset" */
#define SEEK_END  2      /* Set file pointer to EOF plus "offset" */
```

```
/*Pathnames:*/
```

```
#define GF_PATH  /etc/group /*Pathname of the group file */
#define PF_PATH  /etc/passwd/*Pathname of the passwd file */
```



NAME

utmp, wtmp – utmp and wtmp entry formats

SYNOPSIS

```
#include <sys/types.h>
#include <utmp.h>
```

DESCRIPTION

These files, which hold user and accounting information for such commands as *who(1)*, *write(1)*, and *login(1)*, have the following structure as defined by `<utmp.h>`:

```
#define  UTMP_FILE  "/etc/utmp"
#define  WTMP_FILE  "/etc/wtmp"
#define  ut_name    ut_user
```

```
struct utmp {
    char    ut_user[8];          /* User login name */
    char    ut_id[4];           /* /etc/inittab id (usually line #) */
    char    ut_line[12];        /* device name (console, lnxx) */
    short   ut_pid;             /* process id */
    short   ut_type;            /* type of entry */
    struct  exit_status {
        short   e_termination; /* Process termination status */
        short   e_exit;         /* Process exit status */
    } ut_exit;                  /* The exit status of a process
    * marked as DEAD_PROCESS. */
    time_t  ut_time;           /* time entry was made */
};

/* Definitions for ut_type */
#define  EMPTY          0
#define  RUN_LVL        1
#define  BOOT_TIME      2
#define  OLD_TIME       3
#define  NEW_TIME       4
#define  INIT_PROCESS   5          /* Process spawned by "init" */
#define  LOGIN_PROCESS  6          /* A "getty" process waiting for login */
#define  USER_PROCESS   7          /* A user process */
#define  DEAD_PROCESS   8
#define  ACCOUNTING     9
#define  UTMAXTYPE      ACCOUNTING /* Largest legal value of ut_type */
```

UTMP(4)

UTMP(4)

```
/* Special strings or formats used in the "ut_line" field when */
/* accounting for something other than a process */
/* No string for the ut_line field can be more than 11 chars + */
/* a NULL in length */
#define RUNLVL_MSG "run-level %c"
#define BOOT_MSG "system boot"
#define OTIME_MSG "old time"
#define NTIME_MSG "new time"
```

FILES

```
/etc/utmp
/etc/wtmp
```

SEE ALSO

getut(3C) in the *Programmer's Reference Manual*.
login(1), who(1), write(1) in the *User's Reference Manual*.

INTRO(5)

INTRO(5)

NAME

intro – introduction to miscellany

DESCRIPTION

This section describes miscellaneous facilities such as macro packages, character set tables, etc.



NAME

ascii – map of ASCII character set

DESCRIPTION

ascii is a map of the ASCII character set, giving both octal and hexadecimal equivalents of each character, to be printed as needed. It contains:

000 nul	001 soh	002 stx	003 etx	004 eot	005 enq	006 ack	007 bel
010 bs	011 ht	012 nl	013 vt	014 np	015 cr	016 so	017 si
020 dle	021 dc1	022 dc2	023 dc3	024 dc4	025 nak	026 syn	027 etb
030 can	031 em	032 sub	033 esc	034 fs	035 gs	036 rs	037 us
040 sp	041 !	042 "	043 #	044 \$	045 %	046 &	047
050 (051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [134 \	135]	136 ^	137 _
140`	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174	175 }	176 ~	177 del

00 nul	01 soh	02 stx	03 etx	04 eot	05 enq	06 ack	07 bel
08 bs	09 ht	0a nl	0b vt	0c np	0d cr	0e so	0f si
10 dle	11 dc1	12 dc2	13 dc3	14 dc4	15 nak	16 syn	17 etb
18 can	19 em	1a sub	1b esc	1c fs	1d gs	1e rs	1f us
20 sp	21 !	22 "	23 #	24 \$	25 %	26 &	27
28 (29)	2a *	2b +	2c ,	2d -	2e .	2f /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3a :	3b ;	3c <	3d =	3e >	3f ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4a J	4b K	4c L	4d M	4e N	4f O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5a Z	5b [5c \	5d]	5e ^	5f _
60`	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6a j	6b k	6c l	6d m	6e n	6f o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7a z	7b {	7c	7d }	7e ~	7f del



NAME

environ – user environment

DESCRIPTION

An array of strings called the “environment” is made available by *exec(2)* when a process begins. By convention, these strings have the form “name=value”. The following names are used by various commands:

CFTIME The default format string to be used by the *date(1)* command and the *asctime()* and *cftime()* routines (see *ctime(3C)*). If **CFTIME** is not set or is null, the default format string specified in the */lib/cftime/LANGUAGE* file (if it exists) is used in its place (see *cftime(4)*).

CHRCLASS A value that corresponds to a file in */lib/chrclass* containing character classification and conversion information. This information is used

by commands (such as *cat(1)*, *ed(1)*, *sort(1)*, etc.) to classify characters as alphabetic, printable, upper case, etc. and to convert characters to upper or lower case.

When a program or command begins execution, the tables containing this information are initialized based on the value of **CHRCLASS**. If **CHRCLASS** is non-existent, null, set to a value for which no file exists in */lib/chrclass*, or errors occur while reading the file, the ASCII character set is used. During execution, a program or command can change the values in these tables by calling the *setchrclass()* routine. For more detail, see *ctype(3C)*.

These tables are created using the *chrtbl(1M)* command.

HOME The name of the user’s login directory, set by *login(1)* from the password file (see *passwd(4)*).

LANGUAGE A language for which a printable file by that name exists in */lib/cftime*. This information is used by commands (such as *date(1)*, *ls(1)*, *sort(1)*, etc.) to print date and time information in the language specified.

If **LANGUAGE** is non-existent, null, set to a value for which no file exists in */lib/cftime*, or errors occur while reading the file, the last language requested will be used. (If no language has been requested, the language *usa_english* is assumed.) For a description of the content of files in */lib/cftime*, see *cftime(4)*.

PATH The sequence of directory prefixes that *sh(1)*, *time(1)*, *nice(1)*, *nohup(1)*, etc., apply in searching for a file known by an incomplete path name. The prefixes are separated by colons (:). *login(1)* sets **PATH**=*/bin:/usr/bin*. (For more detail, see the “Execution” section of the *sh(1)* manual page.)

TERM The kind of terminal for which output is to be prepared. This information is used by commands, such as *mm(1)* or *vi(1)*, which may exploit special capabilities of that terminal.

TZ Time zone information. The simplest format is `xxxnzzz` where `xxx` is the standard local time zone abbreviation, `n` is the difference in hours from GMT (Greenwich Mean Time), and `zzz` is the abbreviation for an alternate time zone (usually the daylight-saving local time zone), if any; for example, `TZ='EST5EDT'`

The most complex format allows you to specify the difference in hours of the alternate time zone from GMT and the starting day and time and ending day and time for using this alternate time zone. For example, in 1985 the complex format corresponding to the simple example is:

```
TZ='EST5:00:00EDT4:00:00;118/2:00:00,300/2:00:00'
```

For more details, see *ctime*(3C).

Further names may be placed in the environment by the *export* command and "name=value" arguments in *sh*(1), or by *exec*(2). It is unwise to conflict with certain shell variables that are frequently exported by .profile files: MAIL, PS1, PS2, IFS (see *profile*(4)).

NOTES

References to the *cftime*(4), *ctime*(3C), and *ctype*(3C) manual pages refer to programming capabilities available beginning with Issue 4.1 of the C Programming Language Utilities.

Administrators should note the following: if you attempt to set the current date to one of the dates that the standard and alternate time zones change (for example, the date that daylight time is starting or ending), and you attempt to set the time to a time in the interval between the end of standard time and the beginning of the alternate time (or the end of the alternate time and the beginning of standard time), the results are unpredictable.

SEE ALSO

chrtbl(1M), *cftime*(4), *passwd*(4), *profile*(4) in the *System Administrator's Reference Manual*.

exec(2), *ctime*(3C), *ctype*(3C) in the *Programmer's Reference Manual*.

cat(1), *date*(1), *ed*(1), *env*(1), *ls*(1), *login*(1), *nice*(1), *nohup*(1), *sh*(1), *sort*(1), *time*(1), *vi*(1) in the *User's Reference Manual*.

mm(1) in the *DOCUMENTER'S WORKBENCH Software Release 2.0 Technical Discussion and Reference Manual*.

NAME

fcntl – file control options

SYNOPSIS

```
#include <fcntl.h>
```

DESCRIPTION

The *fcntl(2)* function provides for control over open files. This include file describes *requests* and *arguments* to *fcntl* and *open(2)*.

```
/* Flag values accessible to open(2) and fcntl(2) */
```

```
/* (The first three can only be set by open) */
```

```
#define O_RDONLY 0
```

```
#define O_WRONLY 1
```

```
#define O_RDWR 2
```

```
#define O_NDELAY 04 /* Non-blocking I/O */
```

```
#define O_APPEND 010 /* append (writes guaranteed at the end) */
```

```
#define O_SYNC 020 /* synchronous write option */
```

```
/* Flag values accessible only to open(2) */
```

```
#define O_CREAT 00400 /* open with file create (uses third open arg)*/
```

```
#define O_TRUNC 01000 /* open with truncation */
```

```
#define O_EXCL 02000 /* exclusive open */
```

```
/* fcntl(2) requests */
```

```
#define F_DUPFD 0 /* Duplicate files */
```

```
#define F_GETFD 1 /* Get files flags */
```

```
#define F_SETFD 2 /* Set files flags */
```

```
#define F_GETFL 3 /* Get file flags */
```

```
#define F_SETFL 4 /* Set file flags */
```

```
#define F_GETLK 5 /* Get file lock */
```

```
#define F_SETLK 6 /* Set file lock */
```

```
#define F_SETLKW 7 /* Set file lock and wait */
```

```
/* file segment locking control structure */
```

```
struct flock {
    short l_type;
    short l_whence;
    long l_start;
    long l_len; /* if 0 then until EOF */
    short l_sysid; /* returned with F_GETLK*/
    short l_pid; /* returned with F_GETLK*/
}
```

```
/* file segment locking types */
```

```
#define F_RDLCK 01 /* Read lock */
```

FCNTL(5)

FCNTL(5)

```
#define F_WRLCK 02 /* Write lock */  
#define F_UNLCK 03 /* Remove locks */
```

SEE ALSO

fcntl(2), *open(2)* in the *Programmer's Reference Manual*.

NAME

jagent – host control of windowing terminal

SYNOPSIS

```
#include <sys/jioctl.h>
```

```
ioctl (cntlfd, JAGENT, &arg)
```

```
int cntlfd
struct bagent arg
```

DESCRIPTION

The *ioctl(2)* system call, when performed on an *xt(7)* device with the JAGENT request, allows a host program to send information to a windowing terminal.

ioctl has three arguments:

cntlfd the *xt(7)* control channel file descriptor

JAGENT the *xt(7)* *ioctl(2)* request to invoke a windowing terminal agent routine.

arg the address of a *bagent* structure, defined in *<sys/jioctl.h>* as follows:

```
struct      bagent {
    int      size; /* size of src in & dest out */
    char     *src; /* the source byte string */
    char     *dest; /* the destination byte string */
};
```

The *src* pointer must be initialized to point to a byte string which is sent to the windowing terminal. See *layers(5)* for a list of JAGENT strings recognized by windowing terminals. Likewise, the *dest* pointer must be initialized to the address of a buffer to receive a byte string returned by the terminal. When *ioctl(2)* is called, the *size* argument must be set to the length of the *src* string. Upon return, *size* is set by *ioctl(2)* to the length of the destination byte string, *dest*.

DIAGNOSTICS

Upon successful completion, the size of the destination byte string is returned. If an error occurs, *-1* is returned.

SEE ALSO

libwindows(3X), *layers(5)*, *xt(7)*.

ioctl(2) in the *Programmer's Reference Manual*.



NAME

layers – protocol used between host and windowing terminal under *layers*(1)

SYNOPSIS

```
#include <sys/jioctl.h>
```

DESCRIPTION

layers are asynchronous windows supported by the operating system in a windowing terminal. Communication between the UNIX system processes and terminal processes under *layers*(1) occurs via multiplexed channels managed by the respective operating systems using a protocol as specified in *xtproto*(5).

The contents of packets transferring data between a UNIX system process and a layer are asymmetric. Data sent from the UNIX system to a particular terminal process is undifferentiated and it is up to the terminal process to interpret the contents of packets.

Control information for terminal processes is sent via channel 0. Process 0 in the windowing terminal performs the designated functions on behalf of the process connected to the designated channel. These packets take the form:

```
command, channel
```

except for *timeout* and *jagent* information which take the form

```
command, data...
```

The commands are the bottom eight bits extracted from the following *ioctl*(2) codes:

- JBOOT** Prepare to load a new terminal program into the designated layer.
- JTERM** Kill the downloaded layer program, and restore the default window program.
- JTIMO** Set the timeout parameters for the protocol. The data consist of two bytes: the value of the receive timeout in seconds, and the value of the transmit timeout in seconds.
- JTIMOM** Set the timeout parameters for the protocol. The data consist of four bytes in two groups: the value of the receive timeout in milliseconds (the low eight bits followed by the high eight bits) and the value of the transmit timeout (in the same format).
- JZOMBOOT** Like **JBOOT**, but do not execute the program after loading.
- JAGENT** Send a source byte string to the terminal agent routine and wait for a reply byte string to be returned.

The data are from a *jagent* structure (see *jagent*(5)) and consist of a one-byte size field followed by a two-byte agent command code and parameters. Two-byte integers transmitted as part of an agent command are sent with the high-order byte first. The response from the terminal is generally identical to the command packet,

with the two command bytes replaced by the return code: 0 for success, -1 for failure. Note that the routines in the *libwindows(3X)* library all send parameters in an *agentrect* structure. The agent command codes and their parameters are as follows:

A_NEWLAYER	followed by a two-byte channel number and a rectangle structure (four two-byte coordinates).
A_CURRENT	followed by a two-byte channel number.
A_DELETE	followed by a two-byte channel number.
A_TOP	followed by a two-byte channel number.
A_BOTTOM	followed by a two-byte channel number.
A_MOVE	followed by a two-byte channel number and a point to move to (two two-byte coordinates).
A_RESHAPE	followed by a two-byte channel number and the new rectangle (four two-byte coordinates).
A_NEW	followed by a two-byte channel number and a rectangle structure (four two-byte coordinates).
A_EXIT	no parameters needed.
A_ROMVERSION	no parameters needed. The response packet contains the size byte, two-byte return code, two unused bytes, and the parameter part of the terminal id string (e.g., "8;7;3").

Packets from the windowing terminal to the UNIX system all take the following form:

command, data...

The single-byte commands are as follows:

C_SENDCHAR	Send the next byte to the UNIX system process.
C_NEW	Create a new UNIX system process group for this layer. Remember the window size parameters for this layer. The data for this command is in the form described by the <i>jwinsize</i> structure. The size of the window is specified by two 2-byte integers, sent low byte first.
C_UNBLK	Unblock transmission to this layer. There is no data for this command.
C_DELETE	Delete the UNIX system process group attached to this layer. There is no data for this command.
C_EXIT	Exit. Kill all UNIX system process groups associated with this terminal and terminate the session. There is no data for this command.

- C_DEFUNCT** Layer program has died, send a terminate signal to the UNIX system process groups associated with this terminal. There is no data for this command.
- C_SENDCNCHARS** The rest of the data are characters to be passed to the UNIX system process.
- C_RESHAPE** The layer has been reshaped. Change the window size parameters for this layer. The data takes the same form as for the C_NEW command.

SEE ALSO

libwindows(3X) in the *Programmer's Reference Manual*.
layers(1) in the *User's Reference Manual*.
jagent(5), xproto(5), xt(7) in the *System Administrator's Reference Manual*.



NAME

math – math functions and constants

SYNOPSIS

```
#include <math.h>
```

DESCRIPTION

This file contains declarations of all the functions in the Math Library (described in Section 3M), as well as various functions in the C Library (Section 3C) that return floating-point values.

It defines the structure and constants used by the *matherr*(3M) error-handling mechanisms, including the following constant used as an error-return value:

HUGE The maximum value of a single-precision floating-point number.

The following mathematical constants are defined for user convenience:

M_E	The base of natural logarithms (e).
M_LOG2E	The base-2 logarithm of e .
M_LOG10E	The base-10 logarithm of e .
M_LN2	The natural logarithm of 2.
M_LN10	The natural logarithm of 10.
M_PI	π , the ratio of the circumference of a circle to its diameter.
M_PI_2	$\pi/2$.
M_PI_4	$\pi/4$.
M_1_PI	$1/\pi$.
M_2_PI	$2/\pi$.
M_2_SQRTPI	$2/\sqrt{\pi}$.
M_SQRT2	The positive square root of 2.
M_SQRT1_2	The positive square root of 1/2.

For the definitions of various machine-dependent “constants,” see the description of the *<values.h>* header file.

SEE ALSO

values(5).
intro(3), matherr(3M) in the *Programmer's Reference Manual*.



NAME

prof – profile within a function

SYNOPSIS

```
#define MARK
#include <prof.h>
void MARK (name)
```

DESCRIPTION

MARK will introduce a mark called *name* that will be treated the same as a function entry point. Execution of the mark will add to a counter for that mark, and program-counter time spent will be accounted to the immediately preceding mark or to the function if there are no preceding marks within the active function.

Name may be any combination of numbers or underscores. Each *name* in a single compilation must be unique, but may be the same as any ordinary program symbol.

For marks to be effective, the symbol *MARK* must be defined before the header file *<prof.h>* is included. This may be defined by a preprocessor directive as in the synopsis, or by a command line argument, i.e:

```
cc -p -DMARK foo.c
```

If *MARK* is not defined, the *MARK(name)* statements may be left in the source files containing them and will be ignored.

EXAMPLE

In this example, marks can be used to determine how much time is spent in each loop. Unless this example is compiled with *MARK* defined on the command line, the marks are ignored.

```
#include <prof.h>
foo( )
{
    int i, j;
    .
    .
    .
    MARK(loop1);
    for (i = 0; i < 2000; i++) {
        . . .
    }
    MARK(loop2);
    for (j = 0; j < 2000; j++) {
        . . .
    }
}
```

SEE ALSO

prof(1), profil(2), monitor(3C) in the *Programmer's Reference Manual*.



NAME

regex - regular expression compile and match routines

SYNOPSIS

```
#define INIT <declarations>
#define GETC() <getc code>
#define PEEKC() <peekc code>
#define UNGETC(c) <ungetc code>
#define RETURN(pointer) <return code>
#define ERROR(val) <error code>

#include <regex.h>

char *compile (instring, expbuf, endbuf, eof)
char *instring, *expbuf, *endbuf;
int eof;

int step (string, expbuf)
char *string, *expbuf;

extern char *loc1, *loc2, *locs;
extern int circf, sed, nbra;
```

DESCRIPTION

This page describes general-purpose regular expression matching routines in the form of *ed(1)*, defined in `<regex.h>`. Programs such as *ed(1)*, *sed(1)*, *grep(1)*, *bs(1)*, *expr(1)*, etc., which perform regular expression matching use this source file. In this way, only this file need be changed to maintain regular expression compatibility.

The interface to this file is unpleasantly complex. Programs that include this file must have the following five macros declared before the `"#include <regex.h>"` statement. These macros are used by the *compile* routine.

GETC()	Return the value of the next character in the regular expression pattern. Successive calls to GETC() should return successive characters of the regular expression.
PEEKC()	Return the next character in the regular expression. Successive calls to PEEKC() should return the same character [which should also be the next character returned by GETC()].
UNGETC(c)	Cause the argument <i>c</i> to be returned by the next call to GETC() [and PEEKC()]. No more than one character of pushback is ever needed and this character is guaranteed to be the last character read by GETC(). The value of the macro UNGETC(c) is always ignored.
RETURN(pointer)	This macro is used on normal exit of the <i>compile</i> routine. The value of the argument <i>pointer</i> is a pointer to the character after the last character of the compiled regular expression. This is useful to programs which have memory allocation to manage.

ERROR(*val*) This is the abnormal return from the *compile* routine. The argument *val* is an error number (see table below for meanings). This call should never return.

ERROR	MEANING
11	Range endpoint too large.
16	Bad number.
25	"\digit" out of range.
36	Illegal or missing delimiter.
41	No remembered search string.
42	\(\) imbalance.
43	Too many \(.
44	More than 2 numbers given in \{ \}.
45	} expected after \.
46	First number exceeds second in \{ \}.
49	[] imbalance.
50	Regular expression overflow.

The syntax of the *compile* routine is as follows:

```
compile(instring, expbuf, endbuf, eof)
```

The first parameter *instring* is never used explicitly by the *compile* routine but is useful for programs that pass down different pointers to input characters. It is sometimes used in the INIT declaration (see below). Programs which call functions to input characters or have characters in an external array can pass down a value of ((char *) 0) for this parameter.

The next parameter *expbuf* is a character pointer. It points to the place where the compiled regular expression will be placed.

The parameter *endbuf* is one more than the highest address where the compiled regular expression may be placed. If the compiled expression cannot fit in (*endbuf* - *expbuf*) bytes, a call to ERROR(50) is made.

The parameter *eof* is the character which marks the end of the regular expression. For example, in *ed*(1), this character is usually a /.

Each program that includes this file must have a #define statement for INIT. This definition will be placed right after the declaration for the function *compile* and the opening curly brace ({). It is used for dependent declarations and initializations. Most often it is used to set a register variable to point the beginning of the regular expression so that this register variable can be used in the declarations for GETC(), PEEKC() and UNGETC(). Otherwise it can be used to declare external variables that might be used by GETC(), PEEKC() and UNGETC(). See the example below of the declarations taken from *grep*(1).

There are other functions in this file which perform actual regular expression matching, one of which is the function *step*. The call to *step* is as follows:

```
step(string, expbuf)
```

The first parameter to *step* is a pointer to a string of characters to be checked for a match. This string should be null terminated.

The second parameter *expbuf* is the compiled regular expression which was obtained by a call of the function *compile*.

The function *step* returns non-zero if the given string matches the regular expression, and zero if the expressions do not match. If there is a match, two external character pointers are set as a side effect to the call to *step*. The variable set in *step* is *loc1*. This is a pointer to the first character that matched the regular expression. The variable *loc2*, which is set by the function *advance*, points to the character after the last character that matches the regular expression. Thus if the regular expression matches the entire line, *loc1* will point to the first character of *string* and *loc2* will point to the null at the end of *string*.

Step uses the external variable *circf* which is set by *compile* if the regular expression begins with `^`. If this is set then *step* will try to match the regular expression to the beginning of the string only. If more than one regular expression is to be compiled before the first is executed the value of *circf* should be saved for each compiled expression and *circf* should be set to that saved value before each call to *step*.

The function *advance* is called from *step* with the same arguments as *step*. The purpose of *step* is to step through the *string* argument and call *advance* until *advance* returns non-zero indicating a match or until the end of *string* is reached. If one wants to constrain *string* to the beginning of the line in all cases, *step* need not be called; simply call *advance*.

When *advance* encounters a `*` or `{ \ }` sequence in the regular expression, it will advance its pointer to the string to be matched as far as possible and will recursively call itself trying to match the rest of the string to the rest of the regular expression. As long as there is no match, *advance* will back up along the string until it finds a match or reaches the point in the string that initially matched the `*` or `{ \ }`. It is sometimes desirable to stop this backing up before the initial point in the string is reached. If the external character pointer *locs* is equal to the point in the string at sometime during the backing up process, *advance* will break out of the loop that backs up and will return zero. This is used by *ed(1)* and *sed(1)* for substitutions done globally (not just the first occurrence, but the whole line) so, for example, expressions like *s/y//g* do not loop forever.

The additional external variables *sed* and *nbra* are used for special purposes.

EXAMPLES

The following is an example of how the regular expression macros and calls look from *grep(1)*:

```
#define INIT          register char *sp = instring;
#define GETC()        (*sp++)
#define PEEKC()       (*sp)
#define UNGETC(c)     (---sp)
#define RETURN(c)     return;
#define ERROR(c)      regerr()
```

```
#include <regex.h>
```

```
...
```

```
    (void) compile(*argv, expbuf, &expbuf[ESIZE], '^0');
```

```
...
```

```
    if (step(linebuf, expbuf)  
        succeed());
```

SEE ALSO

ed(1), expr(1), grep(1), sed(1) in the *User's Reference Manual*.

NAME

stat – data returned by stat system call

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
```

DESCRIPTION

The system calls *stat* and *fstat* return data whose structure is defined by this include file. The encoding of the field *st_mode* is defined in this file also.

Structure of the result of *stat*

```
struct  stat
{
    dev_t    st_dev;
    ushort   st_ino;
    ushort   st_mode;
    short    st_nlink;
    ushort   st_uid;
    ushort   st_gid;
    dev_t    st_rdev;
    off_t    st_size;
    time_t   st_atime;
    time_t   st_mtime;
    time_t   st_ctime;
};

#define S_IFMT    0170000 /* type of file */
#define S_IFDIR   0040000 /* directory */
#define S_IFCHR   0020000 /* character special */
#define S_IFBLK   0060000 /* block special */
#define S_IFREG   0100000 /* regular */
#define S_IFIFO   0010000 /* fifo */
#define S_ISUID   04000 /* set user id on execution */
#define S_ISGID   02000 /* set group id on execution */
#define S_ISVTX   01000 /* save swapped text even after use */
#define S_IRREAD  00400 /* read permission, owner */
#define S_IWWRITE 00200 /* write permission, owner */
#define S_IXEXEC  00100 /* execute/search permission, owner */
#define S_ENFMT   S_ISGID /* record locking enforcement flag */
#define S_IRWXU   00700 /* read,write, execute: owner */
#define S_IRUSR   00400 /* read permission: owner */
#define S_IWUSR   00200 /* write permission: owner */
#define S_IXUSR   00100 /* execute permission: owner */
#define S_IRWXG   00070 /* read, write, execute: group */
#define S_IRGRP   00040 /* read permission: group */
#define S_IWGRP   00020 /* write permission: group */
#define S_IXGRP   00010 /* execute permission: group */
```

STAT(5)

```
#define S_IRWXO 00007 /* read, write, execute: other */
#define S_IROTH 00004 /* read permission: other */
#define S_IWOTH 00002 /* write permission: other */
#define S_IXOTH 00001 /* execute permission: other */
```

SEE ALSO

types(5).
stat(2) in the *Programmer's Reference Manual*.

STAT(5)

NAME

term – conventional names for terminals

DESCRIPTION

These names are used by certain commands (e.g., *man*(1), *tabs*(1), *tput*(1), *vi*(1) and *curses*(3X)) and are maintained as part of the shell environment in the environment variable **TERM** (see *sh*(1), *profile*(4), and *environ*(5)).

Entries in *terminfo*(4) source files consist of a number of comma-separated fields. (To obtain the source description for a terminal, use the **-I** option of *infocmp*(1M).) White space after each comma is ignored. The first line of each terminal description in the *terminfo*(4) database gives the names by which *terminfo*(4) knows the terminal, separated by bar (|) characters. The first name given is the most common abbreviation for the terminal (this is the one to use to set the environment variable **TERMINFO** in *\$HOME/.profile*; see *profile*(4)), the last name given should be a long name fully identifying the terminal, and all others are understood as synonyms for the terminal name. All names but the last should contain no blanks and must be unique in the first 14 characters; the last name may contain blanks for readability.

Terminal names (except for the last, verbose entry) should be chosen using the following conventions. The particular piece of hardware making up the terminal should have a root name chosen, for example, for the AT&T 4425 terminal, **att4425**. This name should not contain hyphens, except that synonyms may be chosen that do not conflict with other names. Up to 8 characters, chosen from [a-z0-9], make up a basic terminal name. Names should generally be based on original vendors, rather than local distributors. A terminal acquired from one vendor should not have more than one distinct basic name. Terminal sub-models, operational modes that the hardware can be in, or user preferences, should be indicated by appending a hyphen and an indicator of the mode. Thus, an AT&T 4425 terminal in 132 column mode would be **att4425-w**. The following suffixes should be used where possible:

Suffix	Meaning	Example
-w	Wide mode (more than 80 columns)	att4425-w
-am	With auto. margins (usually default)	vt100-am
-nam	Without automatic margins	vt100-nam
-n	Number of lines on the screen	aaa-60
-na	No arrow keys (leave them in local)	c100-na
-np	Number of pages of memory	c100-4p
-rv	Reverse video	att4415-rv

To avoid conflicts with the naming conventions used in describing the different modes of a terminal (e.g., **-w**), it is recommended that a terminal's root name not contain hyphens. Further, it is good practice to make all terminal names used in the *terminfo*(4) database unique. Terminal entries that are present only for inclusion in other entries via the **use=** facilities should have a '+' in their name, as in **4415+nl**.

Some of the known terminal names may include the following (for a complete list, type: **ls -C /usr/lib/terminfo/?**):

TERM(5)

2621, hp2621
 2631
 2631-c
 2631-e
 2640, hp2640
 2645, hp2645
 3270
 33, tty33
 35, tty35
 37, tty37
 4000a
 4014, tek4014
 40, tty40
 43, tty43
 4410, 5410
 4410-nfk, 5410-nfk
 4410-nsl, 5410-nsl
 4410-w, 5410-w
 4410v1, 5410v1
 4410v1-w, 5410v1-w
 4415, 5420
 4415-nl, 5420-nl
 4415-rv, 5420-rv
 4415-rv-nl, 5420-rv-nl
 4415-w, 5420-w
 4415-w-nl, 5420-w-nl

 4415-w-rv, 5420-w-rv
 4415-w-rv-nl, 5420-w-rv-nl

 4418, 5418
 4418-w, 5418-w
 4420
 4424
 4424-2
 4425, 5425
 4425-fk, 5425-fk
 4425-nl, 5425-nl

 4425-w, 5425-w
 4425-w-fk, 5425-w-fk

 4425-nl-w, 5425-nl-w

 4426
 450
 450-12

Hewlett-Packard 2621 series
 Hewlett-Packard 2631 line printer
 Hewlett-Packard 2631 line printer - compressed mode
 Hewlett-Packard 2631 line printer - expanded mode
 Hewlett-Packard 2640 series
 Hewlett-Packard 2645 series
 IBM Model 3270
 AT&T Teletype Model 33 KSR
 AT&T Teletype Model 35 KSR
 AT&T Teletype Model 37 KSR
 Trendata 4000a
 TEKTRONIX 4014
 AT&T Teletype Dataspeed 40/2
 AT&T Teletype Model 43 KSR
 AT&T 4410/5410 terminal in 80-column mode - version 2
 AT&T 4410/5410 without function keys - version 1
 AT&T 4410/5410 without pln defined
 AT&T 4410/5410 in 132-column mode
 AT&T 4410/5410 terminal in 80-column mode - version 1
 AT&T 4410/5410 terminal in 132-column mode - version 1
 AT&T 4415/5420 in 80-column mode
 AT&T 4415/5420 without changing labels
 AT&T 4415/5420 80 columns in reverse video
 AT&T 4415/5420 reverse video without changing labels
 AT&T 4415/5420 in 132-column mode
 AT&T 4415/5420 in 132-column mode without changing labels
 AT&T 4415/5420 132 columns in reverse video
 AT&T 4415/5420 132 columns reverse video without changing labels
 AT&T 5418 in 80-column mode
 AT&T 5418 in 132-column mode
 AT&T Teletype Model 4420
 AT&T Teletype Model 4424
 AT&T Teletype Model 4424 in display function group ii
 AT&T 4425/5425
 AT&T 4425/5425 without function keys
 AT&T 4425/5425 without changing labels in 80-column mode
 AT&T 4425/5425 in 132-column mode
 AT&T 4425/5425 without function keys in 132-column mode
 AT&T 4425/5425 without changing labels in 132-column mode
 AT&T Teletype Model 4426S
 DASI 450 (same as Diablo 1620)
 DASI 450 in 12-pitch mode

500,att500	AT&T-IS 500 terminal
510,510a	AT&T 510/510a in 80-column mode
513bct,att513	AT&T 513 bct terminal
5320	AT&T 5320 hardcopy terminal
5420_2	AT&T 5420 model 2 in 80-column mode
5420_2-w	AT&T 5420 model 2 in 132-column mode
5620,dmd	AT&T 5620 terminal 88 columns
5620-24,dmd-24	AT&T Teletype Model DMD 5620 in a 24x80 layer
5620-34,dmd-34	AT&T Teletype Model DMD 5620 in a 34x80 layer
610,610bct	AT&T 610 bct terminal in 80-column mode
610-w,610bct-w	AT&T 610 bct terminal in 132-column mode
7300,pc7300,unix_pc	AT&T UNIX PC Model 7300
735,ti	Texas Instruments TI735 and TI725
745	Texas Instruments TI745
dumb	generic name for terminals that lack reverse line-feed and other special escape sequences
hp	Hewlett-Packard (same as 2645)
lp	generic name for a line printer
pt505	AT&T Personal Terminal 505 (22 lines)
pt505-24	AT&T Personal Terminal 505 (24-line mode)
sync	generic name for synchronous Teletype Model 4540-compatible terminals

Commands whose behavior depends on the type of terminal should accept arguments of the form `-Tterm` where *term* is one of the names given above; if no such argument is present, such commands should obtain the terminal type from the environment variable `TERM`, which, in turn, should contain *term*.

FILES

`/usr/lib/terminfo/?/*` compiled terminal description database

SEE ALSO

`infocmp(1M)`, `profile(4)`, `terminfo(4)`, `environ(5)` in the *System Administrator's Reference Manual*.

`man(1)`, `sh(1)`, `stty(1)`, `tabs(1)`, `tput(1)`, `tplot(1G)`, `vi(1)` in the *User's Reference Manual*.

`curses(3X)` in the *Programmer's Reference Manual*.

Chapter 10 of the *Programmer's Guide*.

NOTES

Not all programs follow the above naming conventions.



NAME

types – primitive system data types

SYNOPSIS

```
#include <sys/types.h>
```

DESCRIPTION

The data types defined in the include file are used in UNIX system code; some data of these types are accessible to user code:

```
typedef struct { int r[1]; } *physadr;
typedef long      daddr_t;
typedef char *    caddr_t;
typedef unsigned char  unchar;
typedef unsigned short ushort;
typedef unsigned int   uint;
typedef unsigned long  ulong;
typedef ushort        ino_t;
typedef short         cnt_t;
typedef long          time_t;
typedef int           label_t[10];
typedef short         dev_t;
typedef long          off_t;
typedef long          paddr_t;
typedef int           key_t;
typedef unsigned char use_t;
typedef short         sysid_t;
typedef short         index_t;
typedef short         lock_t;
typedef unsigned int  size_t;
```

The form *daddr_t* is used for disk addresses except in an i-node on disk, see *fs(4)*. Times are encoded in seconds since 00:00:00 GMT, January 1, 1970. The major and minor parts of a device code specify kind and unit number of a device and are installation-dependent. Offsets are measured in bytes from the beginning of a file. The *label_t* variables are used to save the processor state while another process is running.

SEE ALSO

fs(4).



NAME

values – machine-dependent values

SYNOPSIS

```
#include <values.h>
```

DESCRIPTION

This file contains a set of manifest constants, conditionally defined for particular processor architectures.

The model assumed for integers is binary representation (one's or two's complement), where the sign is represented by the value of the high-order bit.

BITS(<i>type</i>)	The number of bits in a specified type (e.g., int).
HIBITS	The value of a short integer with only the high-order bit set (in most implementations, 0x8000).
HIBITL	The value of a long integer with only the high-order bit set (in most implementations, 0x80000000).
HIBITI	The value of a regular integer with only the high-order bit set (usually the same as HIBITS or HIBITL).
MAXSHORT	The maximum value of a signed short integer (in most implementations, 0x7FFF \equiv 32767).
MAXLONG	The maximum value of a signed long integer (in most implementations, 0x7FFFFFFF \equiv 2147483647).
MAXINT	The maximum value of a signed regular integer (usually the same as MAXSHORT or MAXLONG).
MAXFLOAT, LN_MAXFLOAT	The maximum value of a single-precision floating-point number, and its natural logarithm.
MAXDOUBLE, LN_MAXDOUBLE	The maximum value of a double-precision floating-point number, and its natural logarithm.
MINFLOAT, LN_MINFLOAT	The minimum positive value of a single-precision floating-point number, and its natural logarithm.
MINDOUBLE, LN_MINDOUBLE	The minimum positive value of a double-precision floating-point number, and its natural logarithm.
FSIGNIF	The number of significant bits in the mantissa of a single-precision floating-point number.
DSIGNIF	The number of significant bits in the mantissa of a double-precision floating-point number.

SEE ALSO

math(5).
intro(3) in the *Programmer's Reference Manual*.



NAME

`varargs` – handle variable argument list

SYNOPSIS

```
#include <varargs.h>
va_alist
va_dcl
void va_start(pvar)
va_list pvar;
type va_arg(pvar, type)
va_list pvar;
void va_end(pvar)
va_list pvar;
```

DESCRIPTION

This set of macros allows portable procedures that accept variable argument lists to be written. Routines that have variable argument lists [such as `printf(3S)`] but do not use `varargs` are inherently nonportable, as different machines use different argument-passing conventions.

`va_alist` is used as the parameter list in a function header.

`va_dcl` is a declaration for `va_alist`. No semicolon should follow `va_dcl`.

`va_list` is a type defined for the variable used to traverse the list.

`va_start` is called to initialize `pvar` to the beginning of the list.

`va_arg` will return the next argument in the list pointed to by `pvar`. *Type* is the type the argument is expected to be. Different types can be mixed, but it is up to the routine to know what type of argument is expected, as it cannot be determined at runtime.

`va_end` is used to clean up.

Multiple traversals, each bracketed by `va_start ... va_end`, are possible.

EXAMPLE

This example is a possible implementation of `execl(2)`.

```
#include <varargs.h>
#define MAXARGS 100

/*     execl is called by
        execl(file, arg1, arg2, ..., (char *)0);
*/
execl(va_alist)
va_dcl
{
    va_list ap;
    char *file;
    char *args[MAXARGS];
    int argno = 0;
```

```
    va_start(ap);
    file = va_arg(ap, char *);
    while ((args[argno++] = va_arg(ap, char *)) != (char *)0)
        ;
    va_end(ap);
    return execv(file, args);
}
```

SEE ALSO

exec(2), *printf(3S)*, *vprintf(3S)* in the *Programmer's Reference Manual*.

NOTES

It is up to the calling routine to specify how many arguments there are, since it is not always possible to determine this from the stack frame. For example, *execl* is passed a zero pointer to signal the end of the list. *Printf* can tell how many arguments are there by the format.

It is non-portable to specify a second argument of *char*, *short*, or *float* to *va_arg*, since arguments seen by the called function are not *char*, *short*, or *float*. C converts *char* and *short* arguments to *int* and converts *float* arguments to *double* before passing them to a function.

NAME

xtproto – multiplexed channels protocol used by xt(7) driver

DESCRIPTION

The xt(7) driver contains routines which implement a multiplexed, multi-buffered, full-duplex protocol with guaranteed delivery of ordered data via an 8-bit byte data stream. This protocol is used for communication between multiple UNIX system host processes and an AT&T windowing terminal operating under layers(1).

The protocol uses packets with a 2-byte header containing a 3-bit sequence number, 3-bit channel number, control flag, and data size. The data part of a packet may not be larger than 32 bytes. The trailer contains a CRC-16 code in 2 bytes. Each channel is double-buffered.

Correctly received packets in sequence are acknowledged with a control packet containing an ACK; however, out of sequence packets generate a control packet containing a NAK, which will cause the retransmission in sequence of all unacknowledged packets.

Unacknowledged packets are retransmitted after a timeout interval which is dependent on baud rate. Another timeout parameter specifies the interval after which incomplete receive packets are discarded.

FILES

/usr/include/sys/xtproto.h channel multiplexing protocol definitions

SEE ALSO

layers(5), xt(7).
layers(1) in the *User's Reference Manual*.



NAME

intro – introduction to special files

DESCRIPTION

This section describes various special files that refer to specific hardware peripherals, and UNIX system device drivers. STREAMS [see *intro(2)*] software drivers, modules and the STREAMS-generic set of *ioctl(2)* system calls are also described.

For hardware related files, the names of the entries are generally derived from names for the hardware, as opposed to the names of the special files themselves. Characteristics of both the hardware device and the corresponding UNIX system device driver are discussed where applicable.

Disk device file names are in the following format:

`/dev/{r}dsk/c#d#s#`

where **r** indicates a raw interface to the disk, the **c#** indicates the controller number, **d#** indicates the device attached to the controller and **s#** indicates the section number of the partitioned device.

SEE ALSO

Disk/Tape Management in the System Administrator's Guide.



NAME

clone – open any minor device on a STREAMS driver

DESCRIPTION

clone is a STREAMS software driver that finds and opens an unused minor device on another STREAMS driver. The minor device passed to *clone* during the open is interpreted as the major device number of another STREAMS driver for which an unused minor device is to be obtained. Each such open results in a separate *stream* to a previously unused minor device.

The *clone* driver consists solely of an open function. This open function performs all of the necessary work so that subsequent system calls (including *close(2)*) require no further involvement of *clone*.

clone will generate an ENXIO error, without opening the device, if the minor device number provided does not correspond to a valid major device, or if the driver indicated is not a STREAMS driver.

CAVEATS

Multiple opens of the same minor device cannot be done through the *clone* interface. Executing *stat(2)* on the file system node for a cloned device yields a different result from executing *fstat(2)* using a file descriptor obtained from opening the node.

SEE ALSO

log(7).
STREAMS Programmer's Guide.



NAME

console – console interface

DESCRIPTION

The console provides the operator interface to the 3B2 computer.

The file */dev/console* is the system console, and refers to an asynchronous serial data line originating from the system board. This special file implements the features described in *termio(7)*.

The file */dev/contty* refers to a second asynchronous serial data line originating from the system board. This special file implements the features described in *termio(7)*.

FILES

/dev/console
/dev/contty

SEE ALSO

termio(7).



NAME

hdelog – hard disk error log interface file

DESCRIPTION

The file `/dev/hdelog` is a special file that provides access to the disk error logging mechanism, the equipped disk table, and the disk drivers of the equipped disks for doing physical (non-partitioned) disk I/O. It is an internal interface of bad block handling and a few other disk utilities and is not intended to be used directly by users. You must be super-user to use it.

FILES

`/dev/hdelog`

SEE ALSO

`hdeadd(1M)`, `hdefix(1M)`, `hdelogger(1M)`.



NAME

id – 3B2 computer Integral Disk Subsystem

DESCRIPTION

The 3B2 computer integral disk subsystem may consist of one or two units in two sizes; 30M and 72M. The files `/dev/dsk/c0d n s0 ... /dev/dsk/c0d n sF` refer to sections of the drive unit number n . This slicing allows the media to be broken up into more manageable pieces.

The `/dev/dsk` files provide access to the disk via the system's normal buffering mechanism. There is also a "raw" interface which provides for direct transfer of a specified number of bytes between the disk and a location in the user's address space. The names of the raw disk files begin with `/dev/rdsk` and end with a number which selects the same disk section as the corresponding `/dev/dsk` file. In raw I/O the read or write must begin on a word boundary; transfer counts can be as small as a single byte.

FILES

`/dev/dsk*`, `/dev/rdsk*`

SEE ALSO

Appendix A of the *System Administrator's Guide* for tables showing the default disk partitioning of a variety of manufacturers' hard disk units.



NAME

if – 3B2 computer Floppy Disk Subsystem

DESCRIPTION

The 3B2 computer floppy disk subsystem consists of one or more diskette drives. The medium contains 1422 blocks. The files *cn dns0* ... *cn dns7* refer to sections of the floppy disk drive. This slicing allows the media to be broken up into more manageable pieces.

The */dev/dsk* files provide access to the disk via the system's normal buffering mechanism. There is also a "raw" interface which provides for direct transfer of a specified number of bytes between the disk and a location in the user's address space. The names of the raw disk files begin with */dev/rdsk* and end with a number which selects the same disk section as the corresponding */dev/dsk* file. In raw I/O the read or write must begin on a word boundary; transfer counts can be as small as a single byte.

FILES

*/dev/dsk**, */dev/rdsk**

SEE ALSO

Appendix A of the *System Administrator's Guide* for tables showing the default disk partitioning of a variety of manufacturers' hard disk units.



NAME

log – interface to STREAMS error logging and event tracing

DESCRIPTION

log is a STREAMS software device driver that provides an interface for the STREAMS error logging and event tracing processes (*strerr*(1M), *strace*(1M)). *log* presents two separate interfaces: a function call interface in the kernel through which STREAMS drivers and modules submit *log* messages; and a subset of *ioctl*(2) system calls and STREAMS messages for interaction with a user level error logger, a trace logger, or processes that need to submit their own *log* messages.

Kernel Interface

log messages are generated within the kernel by calls to the function *strlog*:

```
strlog(mid, sid, level, flags, fmt, arg1, ...)
short mid, sid;
char level;
ushort flags;
char *fmt;
unsigned arg1;
```

Required definitions are contained in `<sys/strlog.h>` and `<sys/log.h>`. *mid* is the STREAMS module id number for the module or driver submitting the *log* message. *sid* is an internal sub-id number usually used to identify a particular minor device of a driver. *level* is a tracing level that allows for selective screening out of low priority messages from the tracer. *flags* are any combination of SL_ERROR (the message is for the error logger), SL_TRACE (the message is for the tracer), SL_FATAL (advisory notification of a fatal error), and SL_NOTIFY (request that a copy of the message be mailed to the system administrator). *fmt* is a *printf*(3S) style format string, except that %s, %e, %E, %g, and %G conversion specifications are not handled. Up to NLOGARGS (currently 3) numeric or character arguments can be provided.

User Interface

log is opened via the clone interface, `/dev/log`. Each open of `/dev/log` obtains a separate *stream* to *log*. In order to receive *log* messages, a process must first notify *log* whether it is an error logger or trace logger via a STREAMS I_STR *ioctl* call (see below). For the error logger, the I_STR *ioctl* has an *ic_cmd* field of I_ERRLOG, with no accompanying data. For the trace logger, the *ioctl* has an *ic_cmd* field of I_TRCLOG, and must be accompanied by a data buffer containing an array of one or more struct *trace_ids* elements. Each *trace_ids* structure specifies an *mid*, *sid*, and *level* from which message will be accepted. *strlog* will accept messages whose *mid* and *sid* exactly match those in the *trace_ids* structure, and whose level is less than or equal to the level given in the *trace_ids* structure. A value of -1 in any of the fields of the *trace_ids* structure indicates that any value is accepted for that field.

At most one trace logger and one error logger can be active at a time. Once the logger process has identified itself via the *ioctl* call, *log* will begin sending up messages subject to the restrictions noted above. These messages are obtained via the *getmsg*(2) system call. The control part of this message

contains a `log_ctl` structure, which specifies the *mid*, *sid*, *level*, *flags*, time in ticks since boot that the message was submitted, the corresponding time in seconds since Jan. 1, 1970, and a sequence number. The time in seconds since 1970 is provided so that the date and time of the message can be easily computed, and the time in ticks since boot is provided so that the relative timing of *log* messages can be determined.

Different sequence numbers are maintained for the error and trace logging *streams*, and are provided so that gaps in the sequence of messages can be determined (during times of high message traffic some messages may not be delivered by the logger to avoid hogging system resources). The data part of the message contains the unexpanded text of the format string (null terminated), followed by `NLOGARGS` words for the arguments to the format string, aligned on the first word boundary following the format string.

A program may also send a message of the same structure to *log*, even if it is not an error or trace logger. The only fields of the `log_ctl` structure in the control part of the message that are accepted are the `level` and `flags` fields; all other fields are filled in by *log* before being forwarded to the appropriate logger. The data portion must contain a null terminated format string, and any arguments (up to `NLOGARGS`) must be packed one word each, on the next word boundary following the end of the format string.

Attempting to issue an `I_TRCLOG` or `I_ERRLOG` when a logging process of the given type already exists will result in the error `ENXIO` being returned. Similarly, `ENXIO` is returned for `I_TRCLOG` *ioctl*s without any `trace_ids` structures, or for any unrecognized `I_STR` *ioctl* calls. Incorrectly formatted *log* messages sent to the driver by a user process are silently ignored (no error results).

EXAMPLES

Example of `I_ERRLOG` notification.

```
struct strioctl ioc;

ioc.ic_cmd = I_ERRLOG;
ioc.ic_timeout = 0;          /* default timeout (15 secs.) */
ioc.ic_len = 0;
ioc.ic_dp = NULL;

ioctl(log, I_STR, &ioc);
```

Example of `I_TRCLOG` notification.

```
struct trace_ids tid[2];

tid[0].ti_mid = 2;
tid[0].ti_sid = 0;
tid[0].ti_level = 1;

tid[1].ti_mid = 1002;
tid[1].ti_sid = -1;        /* any sub-id will be allowed */
```

```
tid[1].ti_level = -1;    /* any level will be allowed */
```

```
ioc.ic_cmd = I_TRCLOG;
ioc.ic_timeout = 0;
ioc.ic_len = 2 * sizeof(struct trace_ids);
ioc.ic_dp = (char *)tid;
```

```
ioctl(log, I_STR, &ioc);
```

Example of submitting a *log* message (no arguments).

```
struct strbuf ctl, dat;
struct log_ctl lc;
char *message = "Don't forget to pick up some milk on the way home";
```

```
ctl.len = ctl.maxlen = sizeof(lc);
ctl.buf = (char *)&lc;
```

```
dat.len = dat.maxlen = strlen(message);
dat.buf = message;
```

```
lc.level = 0;
lc.flags = SL_ERROR|SL_NOTIFY;
```

```
putmsg(log, &ctl, &dat, 0);
```

FILES

/dev/log, <sys/log.h>, <sys/strlog.h>

SEE ALSO

strace(1M), strerr(1M), clone(7).
intro(2), getmsg(2), putmsg(2) in the *Programmer's Reference Manual*.
STREAMS Programmer's Guide.



NAME

mem, kmem – core memory

DESCRIPTION

The file */dev/mem* is a special file that is an image of the core memory of the computer. It may be used, for example, to examine, and even to patch the system.

Byte addresses in */dev/mem* are interpreted as memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

The file */dev/kmem* is the same as */dev/mem* except that kernel virtual memory rather than physical memory is accessed.

The per-process data for the current process begins at 0x80880000.

FILES

/dev/mem
/dev/kmem

WARNING

Some of */dev/kmem* cannot be read because of write-only addresses or unquipped memory addresses.



NAME

mt – tape interface

DESCRIPTION

The files **mt/ctape?** and **rmt/ctape?** refer to cartridge tape controllers (CTC) and associated tape drives. These special device files are linked to the standard CTC **SA/ctape?** and **rSA/ctape?** files, respectively.

The *finc(1M)*, *frec(1M)*, and *labelit(1M)* commands require these magnetic tape file names to work correctly with the CTC. No other CTC commands require these file names.

FILES

/dev/mt/ctape*
/dev/rmt/ctape*

SEE ALSO

finc(1M), *frec(1M)*, *labelit(1M)*.



NULL(7)

NULL(7)

NAME

null – the null file

DESCRIPTION

Data written on the null special file, */dev/null*, is discarded.

Reads from a null special file always return 0 bytes.

FILES

/dev/null



NAME

ports - 5 line asynchronous interface

DESCRIPTION

Each of the five lines attached to a port behaves as described in *termio(7)*. Each port supports 4 RS232 lines and one parallel Centronics interface. The *termio* structure *c_cflag* options B50, B75, B200, EXTA, and EXTB are not available.

FILES

/dev/tty?? serial interface
/dev/lp? parallel interface

SEE ALSO

termio(7).



NAME

prf – operating system profiler

DESCRIPTION

The special file */dev/prf* provides access to activity information in the operating system. Writing the file loads the measurement facility with text addresses to be monitored. Reading the file returns these addresses and a set of counters indicative of activity between adjacent text addresses.

The recording mechanism is driven by the system clock and samples the program counter at line frequency. Samples that catch the operating system are matched against the stored text addresses and increment corresponding counters for later processing.

The file */dev/prf* is a pseudo-device with no associated hardware.

FILES

/dev/prf

SEE ALSO

profiler(1M).



NAME

SA – devices administered by System Administration

DESCRIPTION

The files in the directories `/dev/SA` (for block devices) and the `/dev/rSA` (for raw devices) are used by System Administration to access the devices on which it operates. For devices that support more than one partition (like disks) the `/dev/(r)SA` entry is linked to the partition that spans the entire device. Not all `/dev/(r)SA` entries are used by all System Administration commands.

FILES

`/dev/SA`
`/dev/rSA`

SEE ALSO

`sysadm(1)` in the *User's Reference Manual*.



NAME

streamio – STREAMS ioctl commands

SYNOPSIS

```
#include <stropts.h>
int ioctl (fildes, command, arg)
int fildes, command;
```

DESCRIPTION

STREAMS [see *intro(2)*] ioctl commands are a subset of *ioctl(2)* system calls which perform a variety of control functions on *streams*. The arguments *command* and *arg* are passed to the file designated by *fildes* and are interpreted by the *stream head*. Certain combinations of these arguments may be passed to a module or driver in the *stream*.

fildes is an open file descriptor that refers to a *stream*. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure.

Since these STREAMS commands are a subset of *ioctl*, they are subject to the errors described there. In addition to those errors, the call will fail with *errno* set to EINVAL, without processing a control function, if the *stream* referenced by *fildes* is linked below a multiplexor, or if *command* is not a valid value for a *stream*.

Also, as described in *ioctl*, STREAMS modules and drivers can detect errors. In this case, the module or driver sends an error message to the *stream head* containing an error value. This causes subsequent system calls to fail with *errno* set to this value.

COMMAND FUNCTIONS

The following *ioctl* commands, with error values indicated, are applicable to all STREAMS files:

I_PUSH Pushes the module whose name is pointed to by *arg* onto the top of the current *stream*, just below the *stream head*. It then calls the open routine of the newly-pushed module. On failure, *errno* is set to one of the following values:

- [EINVAL] Invalid module name.
- [EFAULT] *arg* points outside the allocated address space.
- [ENXIO] Open routine of new module failed.
- [ENXIO] Hangup received on *fildes*.

I_POP Removes the module just below the *stream head* of the *stream* pointed to by *fildes*. *arg* should be 0 in an I_POP request. On failure, *errno* is set to one of the following values:

- [EINVAL] No module present in the *stream*.
- [ENXIO] Hangup received on *fildes*.

- I_LOOK** Retrieves the name of the module just below the *stream head* of the *stream* pointed to by *fildev*, and places it in a null terminated character string pointed at by *arg*. The buffer pointed to by *arg* should be at least `FMNAMESZ+1` bytes long. An `[#include <sys/conf.h>]` declaration is required. On failure, *errno* is set to one of the following values:
- [EFAULT] *arg* points outside the allocated address space.
 - [EINVAL] No module present in *stream*.
- I_FLUSH** This request flushes all input and/or output queues, depending on the value of *arg*. Legal *arg* values are:
- FLUSHR Flush read queues.
 - FLUSHW Flush write queues.
 - FLUSHRW Flush read and write queues.
- On failure, *errno* is set to one of the following values:
- [ENOSR] Unable to allocate buffers for flush message due to insufficient STREAMS memory resources.
 - [EINVAL] Invalid *arg* value.
 - [ENXIO] Hangup received on *fildev*.
- I_SETSIG** Informs the *stream head* that the user wishes the kernel to issue the SIGPOLL signal [see *signal(2)* and *sigset(2)*] when a particular event has occurred on the *stream* associated with *fildev*. I_SETSIG supports an asynchronous processing capability in STREAMS. The value of *arg* is a bitmask that specifies the events for which the user should be signaled. It is the bitwise-OR of any combination of the following constants:
- S_INPUT A non-priority message has arrived on a *stream head* read queue, and no other messages existed on that queue before this message was placed there. This is set even if the message is of zero length.
 - S_HIPRI A priority message is present on the *stream head* read queue. This is set even if the message is of zero length.
 - S_OUTPUT The write queue just below the *stream head* is no longer full. This notifies the user that there is room on the queue for sending (or writing) data downstream.
 - S_MSG A STREAMS signal message that contains the SIGPOLL signal has reached the front of the *stream head* read queue.
- A user process may choose to be signaled only of priority messages by setting the *arg* bitmask to the value S_HIPRI.

Processes that wish to receive SIGPOLL signals must explicitly register to receive them using I_SETSIG. If several processes register to receive this signal for the same event on the same Stream, each process will be signaled when the event occurs.

If the value of *arg* is zero, the calling process will be unregistered and will not receive further SIGPOLL signals. On failure, *errno* is set to one of the following values:

[EINVAL] *arg* value is invalid or *arg* is zero and process is not registered to receive the SIGPOLL signal.

[EAGAIN] Allocation of a data structure to store the signal request failed.

I_GETSIG Returns the events for which the calling process is currently registered to be sent a SIGPOLL signal. The events are returned as a bitmask pointed to by *arg*, where the events are those specified in the description of I_SETSIG above. On failure, *errno* is set to one of the following values:

[EINVAL] Process not registered to receive the SIGPOLL signal.

[EFAULT] *arg* points outside the allocated address space.

I_FIND Compares the names of all modules currently present in the *stream* to the name pointed to by *arg*, and returns 1 if the named module is present in the *stream*. It returns 0 if the named module is not present. On failure, *errno* is set to one of the following values:

[EFAULT] *arg* points outside the allocated address space.

[EINVAL] *arg* does not contain a valid module name.

I_PEEK Allows a user to retrieve the information in the first message on the *stream head* read queue without taking the message off the queue. *arg* points to a *strpeek* structure which contains the following members:

```
struct strbuf  ctlbuf;
struct strbuf  databuf;
long           flags;
```

The *maxlen* field in the *ctlbuf* and *databuf* *strbuf* structures [see *getmsg(2)*] must be set to the number of bytes of control information and/or data information, respectively, to retrieve. If the user sets *flags* to RS_HIPRI, I_PEEK will only look for a priority message on the *stream head* read queue.

I_PEEK returns 1 if a message was retrieved, and returns 0 if no message was found on the *stream head* read queue, or if the RS_HIPRI flag was set in *flags* and a priority message was not present on the *stream head* read queue. It does not wait for a message to arrive. On return, *ctlbuf* specifies information in the control buffer, *databuf* specifies information in the data buffer,

and *flags* contains the value 0 or RS_HIPRI. On failure, *errno* is set to the following value:

[EFAULT] *arg* points, or the buffer area specified in *ctlbuf* or *databuf* is, outside the allocated address space.

[EBADMSG] Queued message to be read is not valid for I_PEEK

I_SRDOPT Sets the read mode using the value of the argument *arg*. Legal *arg* values are:

RNORM Byte-stream mode, the default.

RMSGD Message-discard mode.

RMSGN Message-nondiscard mode.

Read modes are described in *read(2)*. On failure, *errno* is set to the following value:

[EINVAL] *arg* is not one of the above legal values.

I_GRDOPT Returns the current read mode setting in an *int* pointed to by the argument *arg*. Read modes are described in *read(2)*. On failure, *errno* is set to the following value:

[EFAULT] *arg* points outside the allocated address space.

I_NREAD Counts the number of data bytes in data blocks in the first message on the *stream head* read queue, and places this value in the location pointed to by *arg*. The return value for the command is the number of messages on the *stream head* read queue. For example, if zero is returned in *arg*, but the *ioctl* return value is greater than zero, this indicates that a zero-length message is next on the queue. On failure, *errno* is set to the following value:

[EFAULT] *arg* points outside the allocated address space.

I_FDINSERT Creates a message from user specified buffer(s), adds information about another *stream* and sends the message downstream. The message contains a control part and an optional data part. The data and control parts to be sent are distinguished by placement in separate buffers, as described below.

arg points to a *strfdinsert* structure which contains the following members:

```

    struct strbuf  ctlbuf;
    struct strbuf  databuf;
    long          flags;
    int           fildes;
    int           offset;

```

The *len* field in the *ctlbuf strbuf* structure [see *putmsg(2)*] must be set to the size of a pointer plus the number of bytes of control information to be sent with the message. *fildes* in the *strfdinsert* structure specifies the file descriptor of the other *stream*. *offset*, which must be word-aligned, specifies the number of bytes

beyond the beginning of the control buffer where `I_FDINSERT` will store a pointer. This pointer will be the address of the read queue structure of the driver for the *stream* corresponding to *fildev* in the *strfdinsert* structure. The *len* field in the *databuf* structure must be set to the number of bytes of data information to be sent with the message or zero if no data part is to be sent.

flags specifies the type of message to be created. A non-priority message is created if *flags* is set to 0, and a priority message is created if *flags* is set to `RS_HIPRI`. For non-priority messages, `I_FDINSERT` will block if the *stream* write queue is full due to internal flow control conditions. For priority messages, `I_FDINSERT` does not block on this condition. For non-priority messages, `I_FDINSERT` does not block when the write queue is full and `O_NDELAY` is set. Instead, it fails and sets *errno* to `EAGAIN`.

`I_FDINSERT` also blocks, unless prevented by lack of internal resources, waiting for the availability of message blocks in the *stream*, regardless of priority or whether `O_NDELAY` has been specified. No partial message is sent. On failure, *errno* is set to one of the following values:

- [EAGAIN] A non-priority message was specified, the `O_NDELAY` flag is set, and the *stream* write queue is full due to internal flow control conditions.
- [ENOSR] Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.
- [EFAULT] *arg* points, or the buffer area specified in *ctlbuf* or *databuf* is, outside the allocated address space.
- [EINVAL] One of the following: *fildev* in the *strfdinsert* structure is not a valid, open *stream* file descriptor; the size of a pointer plus *offset* is greater than the *len* field for the buffer specified through *ctlptr*; *offset* does not specify a properly-aligned location in the data buffer; an undefined value is stored in *flags*.
- [ENXIO] Hangup received on *fildev* of the *ioctl* call or *fildev* in the *strfdinsert* structure.
- [ERANGE] The *len* field for the buffer specified through *databuf* does not fall within the range specified by the maximum and minimum packet sizes of the top-most *stream* module, or the *len* field for the buffer specified through *databuf* is larger than the maximum configured size of the data part of a message, or the *len* field for the buffer specified through *ctlbuf* is larger than the maximum configured size of the control part of a message.

I_FDINSERT can also fail if an error message was received by the *stream head* of the *stream* corresponding to *files* in the *strfdinsert* structure. In this case, *errno* will be set to the value in the message.

I_STR

Constructs an internal STREAMS ioctl message from the data pointed to by *arg*, and sends that message downstream.

This mechanism is provided to send user *ioctl* requests to downstream modules and drivers. It allows information to be sent with the *ioctl*, and will return to the user any information sent upstream by the downstream recipient. I_STR blocks until the system responds with either a positive or negative acknowledgement message, or until the request "times out" after some period of time. If the request times out, it fails with *errno* set to ETIME.

At most, one I_STR can be active on a *stream*. Further I_STR calls will block until the active I_STR completes at the *stream head*. The default timeout interval for these requests is 15 seconds. The O_NDELAY [see *open(2)*] flag has no effect on this call.

To send requests downstream, *arg* must point to a *striocctl* structure which contains the following members:

```
int    ic_cmd;      /* downstream command */
int    ic_timeout; /* ACK/NAK timeout */
int    ic_len;     /* length of data arg */
char  *ic_dp;     /* ptr to data arg */
```

ic_cmd is the internal ioctl command intended for a downstream module or driver and *ic_timeout* is the number of seconds (-1 = infinite, 0 = use default, >0 = as specified) an I_STR request will wait for acknowledgement before timing out. *ic_len* is the number of bytes in the data argument and *ic_dp* is a pointer to the data argument. The *ic_len* field has two uses: on input, it contains the length of the data argument passed in, and on return from the command, it contains the number of bytes being returned to the user (the buffer pointed to by *ic_dp* should be large enough to contain the maximum amount of data that any module or the driver in the *stream* can return).

The *stream head* will convert the information pointed to by the *striocctl* structure to an internal ioctl command message and send it downstream. On failure, *errno* is set to one of the following values:

- [ENOSR] Unable to allocate buffers for the *ioctl* message due to insufficient STREAMS memory resources.
- [EFAULT] *arg* points, or the buffer area specified by *ic_dp* and *ic_len* (separately for data sent and data returned) is, outside the allocated address space.

- [EINVAL] *ic_len* is less than 0 or *ic_len* is larger than the maximum configured size of the data part of a message or *ic_timeout* is less than -1.
- [ENXIO] Hangup received on *fildev*.
- [ETIME] A downstream *ioctl* timed out before acknowledgement was received.

An *I_STR* can also fail while waiting for an acknowledgement if a message indicating an error or a hangup is received at the *stream head*. In addition, an error code can be returned in the positive or negative acknowledgement message, in the event the *ioctl* command sent downstream fails. For these cases, *I_STR* will fail with *errno* set to the value in the message.

I_SENDFD Requests the *stream* associated with *fildev* to send a message, containing a file pointer, to the *stream head* at the other end of a *stream pipe*. The file pointer corresponds to *arg*, which must be an integer file descriptor.

I_SENDFD converts *arg* into the corresponding system file pointer. It allocates a message block and inserts the file pointer in the block. The user id and group id associated with the sending process are also inserted. This message is placed directly on the read queue [see *intro(2)*] of the *stream head* at the other end of the *stream pipe* to which it is connected. On failure, *errno* is set to one of the following values:

- [EAGAIN] The sending *stream* is unable to allocate a message block to contain the file pointer.
- [EAGAIN] The read queue of the receiving *stream head* is full and cannot accept the message sent by *I_SENDFD*.
- [EBADF] *arg* is not a valid, open file descriptor.
- [EINVAL] *fildev* is not connected to a *stream pipe*.
- [ENXIO] Hangup received on *fildev*.

I_RECVFD Retrieves the file descriptor associated with the message sent by an *I_SENDFD* *ioctl* over a *stream pipe*. *arg* is a pointer to a data buffer large enough to hold an *strrecvfd* data structure containing the following members:

```
int fd;
unsigned short uid;
unsigned short gid;
char fill[8];
```

fd is an integer file descriptor. *uid* and *gid* are the user id and group id, respectively, of the sending *stream*.

If *O_NDELAY* is not set [see *open(2)*], *I_RECVFD* will block until a message is present at the *stream head*. If *O_NDELAY* is set, *I_RECVFD* will fail with *errno* set to *EAGAIN* if no message is present at the *stream head*.

If the message at the *stream head* is a message sent by an `I_SENDFD`, a new user file descriptor is allocated for the file pointer contained in the message. The new file descriptor is placed in the *fd* field of the *strrecvfd* structure. The structure is copied into the user data buffer pointed to by *arg*. On failure, *errno* is set to one of the following values:

- [EAGAIN] A message was not present at the *stream head* read queue, and the `O_NDELAY` flag is set.
- [EBADMSG] The message at the *stream head* read queue was not a message containing a passed file descriptor.
- [EFAULT] *arg* points outside the allocated address space.
- [EMFILE] `NOFILES` file descriptors are currently open.
- [ENXIO] Hangup received on *fildes*.

The following two commands are used for connecting and disconnecting multiplexed STREAMS configurations.

I_LINK Connects two *streams*, where *fildes* is the file descriptor of the *stream* connected to the multiplexing driver, and *arg* is the file descriptor of the *stream* connected to another driver. The *stream* designated by *arg* gets connected below the multiplexing driver. `I_LINK` requires the multiplexing driver to send an acknowledgement message to the *stream head* regarding the linking operation. This call returns a multiplexor ID number (an identifier used to disconnect the multiplexor, see `I_UNLINK`) on success, and a -1 on failure. On failure, *errno* is set to one of the following values:

- [ENXIO] Hangup received on *fildes*.
- [ETIME] Time out before acknowledgement message was received at *stream head*.
- [EAGAIN] Temporarily unable to allocate storage to perform the `I_LINK`.
- [ENOSR] Unable to allocate storage to perform the `I_LINK` due to insufficient STREAMS memory resources.
- [EBADF] *arg* is not a valid, open file descriptor.
- [EINVAL] *fildes stream* does not support multiplexing.
- [EINVAL] *arg* is not a *stream*, or is already linked under a multiplexor.
- [EINVAL] The specified link operation would cause a "cycle" in the resulting configuration; that is, if a given *stream head* is linked into a multiplexing configuration in more than one place.

An `I_LINK` can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error or a hangup is received at the *stream head* of *fildes*. In

addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, `I_LINK` will fail with `errno` set to the value in the message.

I_UNLINK

Disconnects the two *streams* specified by *fildev* and *arg*. *fildev* is the file descriptor of the *stream* connected to the multiplexing driver. *fildev* must correspond to the *stream* on which the `ioctl` `I_LINK` command was issued to link the *stream* below the multiplexing driver. *arg* is the multiplexor ID number that was returned by the `I_LINK`. If *arg* is -1, then all Streams which were linked to *fildev* are disconnected. As in `I_LINK`, this command requires the multiplexing driver to acknowledge the unlink. On failure, `errno` is set to one of the following values:

- [ENXIO] Hangup received on *fildev*.
- [ETIME] Time out before acknowledgement message was received at *stream head*.
- [ENOSR] Unable to allocate storage to perform the `I_UNLINK` due to insufficient STREAMS memory resources.
- [EINVAL] *arg* is an invalid multiplexor ID number or *fildev* is not the *stream* on which the `I_LINK` that returned *arg* was performed.

An `I_UNLINK` can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error or a hangup is received at the *stream head* of *fildev*. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, `I_UNLINK` will fail with `errno` set to the value in the message.

SEE ALSO

`close(2)`, `fcntl(2)`, `intro(2)`, `ioctl(2)`, `open(2)`, `read(2)`, `getmsg(2)`, `poll(2)`, `putmsg(2)`, `signal(2)`, `sigset(2)`, `write(2)` in the *Programmer's Reference Manual*.
STREAMS Programmer's Guide.
STREAMS Primer.

DIAGNOSTICS

Unless specified otherwise above, the return value from `ioctl` is 0 upon success and -1 upon failure with `errno` set as indicated.



NAME

sxt – pseudo-device driver

DESCRIPTION

The special file `/dev/sxt` is a pseudo-device driver that interposes a discipline between the standard `tty` line disciplines and a real device driver. The standard disciplines manipulate *virtual tty* structures (channels) declared by the `/dev/sxt` driver. `/Dev/sxt` acts as a discipline manipulating a *real tty* structure declared by a real device driver. The `/dev/sxt` driver is currently only used by the `shl(1)` command.

Virtual ttys are named by inodes in the subdirectory `/dev/sxt` and are allocated in groups of up to eight. To allocate a group, a program should exclusively open a file with a name of the form `/dev/sxt/??0` (channel 0) and then execute a `SXTIOCLINK ioctl` call to initiate the multiplexing.

Only one channel, the *controlling* channel, can receive input from the keyboard at a time; others attempting to read will be blocked.

There are two groups of `ioctl(2)` commands supported by `sxt`. The first group contains the standard `ioctl` commands described in `termio(7)`, with the addition of the following:

`TIOCEXCL` Set *exclusive use* mode: no further opens are permitted until the file has been closed.

`TIOCNXCL` Reset *exclusive use* mode: further opens are once again permitted.

The second group are commands to `sxt` itself. Some of these may only be executed on channel 0.

`SXTIOCLINK` Allocate a channel group and multiplex the virtual ttys onto the real tty. The argument is the number of channels to allocate. This command may only be executed on channel 0. Possible errors include:

`EINVAL` The argument is out of range.

`ENOTTY` The command was not issued from a real tty.

`ENXIO` *linesw* is not configured with `sxt`.

`EBUSY` An `SXTIOCLINK` command has already been issued for this real tty.

`ENOMEM` There is no system memory available for allocating the virtual tty structures.

`EBADF` Channel 0 was not opened before this call.

`SXTIOCSWTC` Set the controlling channel. Possible errors include:

`EINVAL` An invalid channel number was given.

`EPERM` The command was not executed from channel 0.

SXTIOCWF	Cause a channel to wait until it is the controlling channel. This command will return the error, <i>EINVAL</i> , if an invalid channel number is given.
SXTIOCUBLK	Turn off the <i>lblk</i> control flag in the virtual tty of the indicated channel. The error <i>EINVAL</i> will be returned if an invalid number or channel 0 is given.
SXTIOCSTAT	Get the status (blocked on input or output) of each channel and store in the <i>sxtblock</i> structure referenced by the argument. The error <i>EFAULT</i> will be returned if the structure cannot be written.
SXTIOCTRACE	Enable tracing. Tracing information is written to the console on the 3B2 Computer. This command has no effect if tracing is not configured.
SXTIOCNOTRACE	Disable tracing. This command has no effect if tracing is not configured.

FILES

/dev/sxt/??[0-7] Virtual tty devices

SEE ALSO

termio(7).
shl(1), *stty(1)* in the *User's Reference Manual*.
ioctl(2), *open(2)* in the *Programmer's Reference Manual*.

NAME

termio — general terminal interface

DESCRIPTION

All of the asynchronous communications ports use the same general interface, no matter what hardware is involved. The remainder of this section discusses the common features of this interface.

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users' programs seldom open terminal files; they are opened by *getty* and become a user's standard input, output, and error files. The very first terminal file opened by the process group leader of a terminal file not already associated with a process group becomes the *control terminal* for that process group. The control terminal plays a special role in handling quit and interrupt signals, as discussed below. The control terminal is inherited by a child process during a *fork(2)*. A process can break this association by changing its process group using *setpgrp(2)*.

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely full, which is rare, or when the user has accumulated the maximum allowed number of input characters that have not yet been read by some program. Currently, this limit is 256 characters. When the input limit is reached, the buffer is flushed and all the saved characters are thrown away without notice.

Normally, terminal input is processed in units of lines. A line is delimited by a new-line (ASCII LF) character, an end-of-file (ASCII EOT) character, or an end-of-line character. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, erase and kill processing is normally done. By default, the character # erases the last character typed, except that it will not erase beyond the beginning of the line. By default, the character @ kills (deletes) the entire input line, and optionally outputs a new-line character. Both these characters operate on a key-stroke basis, independently of any backspacing or tabbing that may have been done. Both the erase and kill characters may be entered literally by preceding them with the escape character (\). In this case the escape character is not read. The erase and kill characters may be changed.

Certain characters have special functions on input. These functions and their default character values are summarized as follows:

INTR (Rubout or ASCII DEL) generates an *interrupt* signal which is sent to all processes with the associated control terminal. Normally, each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed-upon location; see *signal(2)*.

- QUIT (Control-| or ASCII FS) generates a *quit* signal. Its treatment is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will not only be terminated but a core image file (called *core*) will be created in the current working directory.
- SWTCH (Control-z or ASCII SUB) is used by the job control facility, *shl*, to change the current layer to the control layer.
- ERASE (#) erases the preceding character. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.
- KILL (@) deletes the entire line, as delimited by a NL, EOF, or EOL character.
- EOF (Control-d or ASCII EOT) may be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line, and the EOF is discarded. Thus, if there are no characters waiting, which is to say the EOF occurred at the beginning of a line, zero characters will be passed back, which is the standard end-of-file indication.
- NL (ASCII LF) is the normal line delimiter. It can not be changed or escaped.
- EOL (ASCII NUL) is an additional line delimiter, like NL. It is not normally used.
- EOL2 is another additional line delimiter.
- STOP (Control-s or ASCII DC3) can be used to temporarily suspend output. It is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended, STOP characters are ignored and not read.
- START (Control-q or ASCII DC1) is used to resume output which has been suspended by a STOP character. While output is not suspended, START characters are ignored and not read. The start/stop characters can not be changed or escaped.

The character values for INTR, QUIT, SWTCH, ERASE, KILL, EOF, and EOL may be changed to suit individual tastes. The ERASE, KILL, and EOF characters may be escaped by a preceding \ character, in which case no special function is done.

When the carrier signal from the data-set drops, a *hang-up* signal is sent to all processes that have this terminal as the control terminal. Unless other arrangements have been made, this signal causes the processes to terminate. If the hang-up signal is ignored, any subsequent read returns with an end-of-file indication. Thus, programs that read a terminal and test for end-of-file can terminate appropriately when hung up on.

When one or more characters are written, they are transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters more rapidly than they can be typed, it will be

suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the program is resumed.

Several *ioctl(2)* system calls apply to terminal files. The primary calls use the following structure, defined in `<termio.h>`:

```
#define NCC      8
struct termio {
    unsigned short c_iflag;    /* input modes */
    unsigned short c_oflag;    /* output modes */
    unsigned short c_cflag;    /* control modes */
    unsigned short c_lflag;    /* local modes */
    char c_line;    /* line discipline */
    unsigned char c_cc[NCC];    /* control chars */
};
```

The special control characters are defined by the array *c_cc*. The relative positions and initial values for each function are as follows:

```
0  VINTR  DEL
1  VQUIT  FS
2  VERASE  #
3  VKILL  @
4  VEOF   EOT
5  VEOL   NUL
6  reserved
7  SWTCH
```

The *c_iflag* field describes the basic terminal input control:

```
IGNBRK  0000001  Ignore break condition.
BRKINT  0000002  Signal interrupt on break.
IGNPAR  0000004  Ignore characters with parity errors.
PARMRK  0000010  Mark parity errors.
INPCK   0000020  Enable input parity check.
ISTRIP  0000040  Strip character.
INLCR   0000100  Map NL to CR on input.
IGNCR   0000200  Ignore CR.
ICRNL   0000400  Map CR to NL on input.
IUCLC   0001000  Map upper-case to lower-case on input.
IXON    0002000  Enable start/stop output control.
IXANY   0004000  Enable any character to restart output.
IXOFF   0010000  Enable start/stop input control.
```

If *IGNBRK* is set, the break condition (a character framing error with data all zeros) is ignored, that is, not put on the input queue and therefore not read by any process. Otherwise if *BRKINT* is set, the break condition will generate an interrupt signal and flush both the input and output queues. If *IGNPAR* is set, characters with other framing and parity errors are ignored.

If *PARMRK* is set, a character with a framing or parity error which is not ignored is read as the three-character sequence: 0377, 0, X, where X is the data of the character received in error. To avoid ambiguity in this case, if *ISTRIP* is not set, a valid character of 0377 is read as 0377, 0377. If *PARMRK* is

not set, a framing or parity error which is not ignored is read as the character NUL (0).

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled. This allows output parity generation without input parity errors.

If ISTRIP is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If INLCR is set, a received NL character is translated into a CR character. If IGNCR is set, a received CR character is ignored (not read). Otherwise if ICRNL is set, a received CR character is translated into a NL character.

If IUCLC is set, a received upper-case alphabetic character is translated into the corresponding lower-case character.

If IXON is set, start/stop output control is enabled. A received STOP character will suspend output and a received START character will restart output. All start/stop characters are ignored and not read. If IXANY is set, any input character, will restart output which has been suspended.

If IXOFF is set, the system will transmit START/STOP characters when the input queue is nearly empty/full.

The initial input control value is all-bits-clear.

The *c_oflag* flag specifies the system treatment of output:

OPOST	0000001	Postprocess output.
OLCUC	0000002	Map lower case to upper on output.
ONLCR	0000004	Map NL to CR-NL on output.
OCRNL	0000010	Map CR to NL on output.
ONOCR	0000020	No CR output at column 0.
ONLRET	0000040	NL performs CR function.
OFILL	0000100	Use fill characters for delay.
OFDEL	0000200	Fill is DEL, else NUL.
NLDLY	0000400	Select new-line delays:
NL0	0	
NL1	0000400	
CRDLY	0003000	Select carriage-return delays:
CR0	0	
CR1	0001000	
CR2	0002000	
CR3	0003000	
TABDLY	0014000	Select horizontal-tab delays:
TAB0	0	
TAB1	0004000	
TAB2	0010000	
TAB3	0014000	Expand tabs to spaces.
BSDLY	0020000	Select backspace delays:
BS0	0	
BS1	0020000	
VTDLY	0040000	Select vertical-tab delays:
VT0	0	

```

VT1      0040000
FFDLY    0100000  Select form-feed delays:
FF0      0
FF1      0100000

```

If OPOST is set, output characters are post-processed as indicated by the remaining flags, otherwise characters are transmitted without change.

If OLCUC is set, a lower-case alphabetic character is transmitted as the corresponding upper-case character. This function is often used in conjunction with IUCLC.

If ONLCR is set, the NL character is transmitted as the CR-NL character pair. If OCRNL is set, the CR character is transmitted as the NL character. If ONOCR is set, no CR character is transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the carriage-return function; the column pointer will be set to 0 and the delays specified for CR will be used. Otherwise the NL character is assumed to do just the line-feed function; the column pointer will remain unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set, fill characters will be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need only a minimal delay. If OFDEL is set, the fill character is DEL, otherwise NUL.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If ONLRET is set, the carriage-return delays are used instead of the new-line delays. If OFILL is set, two fill characters will be transmitted.

Carriage-return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If OFILL is set, delay type 1 transmits two fill characters, and type 2, four fill characters.

Horizontal-tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If OFILL is set, two fill characters will be transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set, one fill character will be transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

The *c_flag* field describes the hardware control of the terminal:

```

CBAUD    0000017  Baud rate:
B0       0        Hang up
B50      0000001  50 baud
B75      0000002  75 baud
B110     0000003  110 baud

```

B134	0000004	134 baud
B150	0000005	150 baud
B200	0000006	200 baud
B300	0000007	300 baud
B600	0000010	600 baud
B1200	0000011	1200 baud
B1800	0000012	1800 baud
B2400	0000013	2400 baud
B4800	0000014	4800 baud
B9600	0000015	9600 baud
B19200	0000016	19200 baud
EXTA	0000016	External A
B38400	0000017	38400 baud
EXTB	0000017	External B
CSIZE	0000060	Character size:
CS5	0	5 bits
CS6	0000020	6 bits
CS7	0000040	7 bits
CS8	0000060	8 bits
CSTOPB	0000100	Send two stop bits, else one.
CREAD	0000200	Enable receiver.
PARENB	0000400	Parity enable.
PARODD	0001000	Odd parity, else even.
HUPCL	0002000	Hang up on last close.
CLOCAL	0004000	Local line, else dial-up.
RCV1EN	0010000	
XMT1EN	0020000	
LOBLK	0040000	Block layer output.

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal will not be asserted. Normally, this will disconnect the line. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used, otherwise one stop bit. For example, at 110 baud, two stop bits are required.

If PARENB is set, parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set, otherwise even parity is used.

If CREAD is set, the receiver is enabled. Otherwise no characters will be received.

If HUPCL is set, the line will be disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal will not be asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control. Otherwise modem control is assumed.

If LOBLK is set, the output of a job control layer will be blocked when it is not the current layer. Otherwise the output generated by that layer will be multiplexed onto the current layer.

The initial hardware control value after open is B300, CS8, CREAD, HUPCL.

The *c_flag* field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline (0) provides the following:

ISIG	0000001	Enable signals.
ICANON	0000002	Canonical input (erase and kill processing).
XCASE	0000004	Canonical upper/lower presentation.
ECHO	0000010	Enable echo.
ECHOE	0000020	Echo erase character as BS-SP-BS.
ECHOK	0000040	Echo NL after kill character.
ECHONL	0000100	Echo NL.
NOFLSH	0000200	Disable flush after interrupt or quit.

If ISIG is set, each input character is checked against the special control characters INTR, SWTCH, and QUIT. If an input character matches one of these control characters, the function associated with that character is performed. If ISIG is not set, no checking is done. Thus these special input functions are possible only if ISIG is set. These functions may be disabled individually by changing the value of the control character to an unlikely or impossible value (e.g., 0377).

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, and EOL. If ICANON is not set, read requests are satisfied directly from the input queue. A read will not be satisfied until at least MIN characters have been received or the timeout value TIME has expired between characters. This allows fast bursts of input to be read efficiently while still allowing single character input. The MIN and TIME values are stored in the position for the EOF and EOL characters, respectively. The time value represents tenths of seconds.

If XCASE is set, and if ICANON is set, an upper-case letter is accepted on input by preceding it with a \ character, and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

<i>for:</i>	<i>use:</i>
\	\'
	!\
{	\{
}	\}
\	\\

For example, A is input as \a, \n as \\n, and \N as \\N.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible. If ECHO and ECHOE are set, the erase character is echoed as ASCII BS SP BS, which will clear the last character from a CRT screen. If ECHOE is set and ECHO is not set, the erase character is echoed as ASCII SP BS. If ECHOK is set, the NL character will be echoed after the kill character to emphasize that the line will be deleted. Note that an escape character preceding the erase or kill character removes any special function. If ECHONL is set, the NL character will be echoed even if ECHO is not set. This is useful for terminals set to local echo (so-called half duplex). Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit, switch, and interrupt characters will not be done.

The initial line-discipline control value is all bits clear.

The primary *ioctl(2)* system calls have the form:

```
ioctl (fildes, command, arg)
struct termio *arg;
```

The commands using this form are:

TCGETA	Get the parameters associated with the terminal and store in the <i>termio</i> structure referenced by <i>arg</i> .
TCSETA	Set the parameters associated with the terminal from the structure referenced by <i>arg</i> . The change is immediate.
TCSETAW	Wait for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.
TCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters.

Additional *ioctl(2)* calls have the form:

```
ioctl (fildes, command, arg)
int arg;
```

The commands using this form are:

TCSBRK	Wait for the output to drain. If <i>arg</i> is 0, then send a break (zero bits for 0.25 seconds).
TCXONC	Start/stop control. If <i>arg</i> is 0, suspend output; if 1, restart suspended output.
TCFLSH	If <i>arg</i> is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues.

FILES

/dev/tty*

SEE ALSO

stty(1) in the *User's Reference Manual*.

fork(2), *ioctl(2)*, *setpgpr(2)*, *signal(2)* in the *Programmer's Reference Manual*.

NAME

`timod` – Transport Interface cooperating STREAMS module

DESCRIPTION

timod is a STREAMS module for use with the Transport Interface (TI) functions of the Network Services library. The *timod* module converts a set of *ioctl(2)* calls into STREAMS messages that may be consumed by a transport protocol provider which supports the Transport Interface. This allows a user to initiate certain TI functions as atomic operations.

The *timod* module must be pushed (see *Streams Primer*) onto only a *stream* terminated by a transport protocol provider which supports the TI.

All STREAMS messages, with the exception of the message types generated from the *ioctl* commands described below, will be transparently passed to the neighboring STREAMS module or driver. The messages generated from the following *ioctl* commands are recognized and processed by the *timod* module. The format of the *ioctl* call is:

```
#include <sys/stropts.h>
-
-
struct strioctl strioctl;
-
-
strioctl.ic_cmd = cmd;
strioctl.ic_timeout = INFTIM;
strioctl.ic_len = size;
strioctl.ic_dp = (char *)buf

ioctl(fildes, I_STR, &strioctl);
```

Where, on issuance, *size* is the size of the appropriate TI message to be sent to the transport provider and on return *size* is the size of the appropriate TI message from the transport provider in response to the issued TI message. *buf* is a pointer to a buffer large enough to hold the contents of the appropriate TI messages. The TI message types are defined in *<sys/tihdr.h>*. The possible values for the *cmd* field are:

- | | |
|-------------------|---|
| TI_BIND | Bind an address to the underlying transport protocol provider. The message issued to the <i>TI_BIND ioctl</i> is equivalent to the TI message type <i>T_BIND_REQ</i> and the message returned by the successful completion of the <i>ioctl</i> is equivalent to the TI message type <i>T_BIND_ACK</i> . |
| TI_UNBIND | Unbind an address from the underlying transport protocol provider. The message issued to the <i>TI_UNBIND ioctl</i> is equivalent to the TI message type <i>T_UNBIND_REQ</i> and the message returned by the successful completion of the <i>ioctl</i> is equivalent to the TI message type <i>T_OK_ACK</i> . |
| TI_GETINFO | Get the TI protocol specific information from the transport protocol provider. The message issued to the <i>TI_GETINFO ioctl</i> is equivalent to the TI message type <i>T_INFO_REQ</i> and the |

message returned by the successful completion of the *ioctl* is equivalent to the TI message type T_INFO_ACK.

TI_OPTMGMT Get, set or negotiate protocol specific options with the transport protocol provider. The message issued to the TI_OPTMGMT *ioctl* is equivalent to the TI message type T_OPTMGMT_REQ and the message returned by the successful completion of the *ioctl* is equivalent to the TI message type T_OPTMGMT_ACK.

FILES

<sys/timod.h>
<sys/tiuser.h>
<sys/tihdr.h>
<sys/errno.h>

SEE ALSO

tirdwr(7).
STREAMS Primer.
STREAMS Programmer's Guide.
Network Programmer's Guide.

DIAGNOSTICS

If the *ioctl* system call returns with a value greater than 0, the lower 8 bits of the return value will be one of the TI error codes as defined in <sys/tiuser.h>. If the TI error is of type TSYSEERR, then the next 8 bits of the return value will contain an error as defined in <sys/errno.h> (see *intro(2)*).

NAME

tirdwr – Transport Interface read/write interface STREAMS module

DESCRIPTION

tirdwr is a STREAMS module that provides an alternate interface to a transport provider which supports the Transport Interface (TI) functions of the Network Services library (see Section 3N). This alternate interface allows a user to communicate with the transport protocol provider using the *read(2)* and *write(2)* system calls. The *putmsg(2)* and *getmsg(2)* system calls may also be used. However, *putmsg* and *getmsg* can only transfer data messages between user and *stream*.

The *tirdwr* module must only be pushed [see *I_PUSH* in *streamio(7)*] onto a *stream* terminated by a transport protocol provider which supports the TI. After the *tirdwr* module has been pushed onto a *stream*, none of the Transport Interface functions can be used. Subsequent calls to TI functions will cause an error on the *stream*. Once the error is detected, subsequent system calls on the *stream* will return an error with *errno* set to *EPROTO*.

The following are the actions taken by the *tirdwr* module when pushed on the *stream*, popped [see *I_POP* in *streamio(7)*] off the *stream*, or when data passes through it.

- push* – When the module is pushed onto a *stream*, it will check any existing data destined for the user to ensure that only regular data messages are present. It will ignore any messages on the *stream* that relate to process management, such as messages that generate signals to the user processes associated with the *stream*. If any other messages are present, the *I_PUSH* will return an error with *errno* set to *EPROTO*.
- write* – The module will take the following actions on data that originated from a *write* system call:
 - All messages with the exception of messages that contain control portions (see the *putmsg* and *getmsg* system calls) will be transparently passed onto the module's downstream neighbor.
 - Any zero length data messages will be freed by the module and they will not be passed onto the module's downstream neighbor.
 - Any messages with control portions will generate an error, and any further system calls associated with the *stream* will fail with *errno* set to *EPROTO*.
- read* – The module will take the following actions on data that originated from the transport protocol provider:
 - All messages with the exception of those that contain control portions (see the *putmsg* and *getmsg* system calls) will be transparently passed onto the module's upstream neighbor.
 - The action taken on messages with control portions will be as follows:

- + Messages that represent expedited data will generate an error. All further system calls associated with the *stream* will fail with *errno* set to EPROTO.
 - + Any data messages with control portions will have the control portions removed from the message prior to passing the message on to the upstream neighbor.
 - + Messages that represent an orderly release indication from the transport provider will generate a zero length data message, indicating the end of file, which will be sent to the reader of the *stream*. The orderly release message itself will be freed by the module.
 - + Messages that represent an abortive disconnect indication from the transport provider will cause all further *write* and *putmsg* system calls to fail with *errno* set to ENXIO. All further *read* and *getmsg* system calls will return zero length data (indicating end of file) once all previous data has been read.
 - + With the exception of the above rules, all other messages with control portions will generate an error and all further system calls associated with the *stream* will fail with *errno* set to EPROTO.
 - Any zero length data messages will be freed by the module and they will not be passed onto the module's upstream neighbor.
- pop* - When the module is popped off the *stream* or the *stream* is closed, the module will take the following action:
- If an orderly release indication has been previously received, then an orderly release request will be sent to the remote side of the transport connection.

SEE ALSO

streamio(7), *timod(7)*.
intro(2), *getmsg(2)*, *putmsg(2)*, *read(2)*, *write(2)*, *intro(3)* in the *Programmer's Reference Manual*.
STREAMS Primer.
STREAMS Programmer's Guide.
Network Programmer's Guide.

NAME

tty – controlling terminal interface

DESCRIPTION

The file */dev/tty* is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

FILES

/dev/tty
*/dev/tty**

SEE ALSO

console(7), *ports(7)*.



NAME

xt – multiplexed tty driver for AT&T windowing terminals

DESCRIPTION

The *xt* driver provides virtual *tty(7)* circuits multiplexed onto real *tty(7)* lines. It interposes its own channel multiplexing protocol as a line discipline between the real device driver and the standard *tty(7)* line disciplines.

Virtual *tty(7)* circuits are named by character-special files of the form */dev/xt???*. Filenames end in three digits, where the first two represent the channel group and the last represents the virtual *tty(7)* number (0-7) of the channel group. Allocation of a new channel group is done dynamically by attempting to open a name ending in 0 with the *O_EXCL* flag set. After a successful open, the *tty(7)* file onto which the channels are to be multiplexed should be passed to *xt* via the *XTIOCLINK ioctl(2)* request. Afterwards, all the channels in the group will behave as normal *tty(7)* files, with data passed in packets via the real *tty(7)* line.

The *xt* driver implements the protocol described in *xtproto(5)* and in *layers(5)*. Packets are formatted as described in *xtproto(5)*, while the contents of packets conform to the description in *layers(5)*.

There are three groups of *ioctl(2)* requests recognized by *xt*. The first group contains all the normal *tty ioctl(2)* requests described in *termio(7)*, with the addition of the following:

TIOCEXCL	Set exclusive use mode; no further opens are permitted until the file has been closed.
TIOCNXCL	Reset exclusive use mode; further opens are once again permitted.

The second group of *ioctl(2)* requests concerns control of the windowing terminal, and is described in the header file *<sys/jioctl.h>*. The requests are as follows:

JTYPE, JMPX	Both return the value <i>JMPX</i> . These are used to identify a terminal device as an <i>xt</i> channel.
JBOOT, JTERM	Both generate an appropriate command packet to the windowing terminal affecting the layer associated with the file descriptor argument to <i>ioctl(2)</i> . They may return the error code <i>EIO</i> if the system <i>clist</i> is empty.
JTIMO, JTIMOM	<i>JTIMO</i> specifies the timeouts in seconds, and <i>JTIMOM</i> in milliseconds. Invalid except on channel 0. They may return the error code <i>EIO</i> if the system <i>clist</i> is empty.
JWINSIZE	Requires the address of a <i>jwinsize</i> structure as an argument. The window sizes of the layer associated with the file descriptor argument to <i>ioctl(2)</i> are copied to the structure.
JZOMBOOT	Generate a command packet to the windowing terminal to enter download mode on the channel associated with the file descriptor argument to <i>ioctl(2)</i> , like <i>JBOOT</i> ; but when the download is finished, make the layer a zombie (ready

for debugging). It may return the error code EIO if the system *clist* is empty.

JAGENT Send the supplied data as a command packet to invoke a windowing terminal agent routine, and return the terminal's response to the calling process. Invalid except on the file descriptor for channel 0. See *jagent(5)*. It may return the error code EIO if the system *clist* is empty.

The third group of *ioctl(2)* requests concerns the configuration of *xt*, and is described in the header file `<sys/xt.h>`. The requests are as follows:

XTIOCTYPE Returns the value XTIOCTYPE.

XTIOCLINK Requires an argument that is a structure, *xtioclm*, containing a file descriptor for the file to be multiplexed and the maximum number of channels allowed. Invalid except on channel 0. This request may return one of the following errors:

- EINVAL** *nchans* has an illegal value.
- ENOTTY** *fd* does not describe a real *tty(7)* device.
- ENXIO** *linesw* is not configured with *xt*.
- EBUSY** An XTIOCLINK request has already been issued for the channel group.
- ENOMEM** There is no system memory available for allocating to the *tty(7)* structures.
- EIO** The JTIMOM packet described above could not be delivered.

HXTIOCLINK Like XTIOCLINK, but specifies that ENCODING MODE be used.

XTIOCTRACE Requires the address of a *Tbuf* structure as an argument. The structure is filled with the contents of the driver trace buffer. Tracing is enabled. This request is invalid if tracing is not configured.

XTIOCNOTRACE Tracing is disabled. This request is invalid if tracing is not configured.

XTIOCSTATS Requires an argument that is the address of an array of size `S_NSTATS`, of type *Stats_t*. The array is filled with the contents of the driver statistics array. This request is invalid if statistics are not configured.

XTIOCADATA Requires the address of a maximum-sized *Link* structure as an argument. The structure is filled with the contents of the driver *Link* data. This request is invalid if data extraction is not configured.

FILES

/dev/xt/??[0-7]	multiplexed special files
/usr/include/sys/jioctl.h	packet command types
/usr/include/sys/xtproto.h	channel multiplexing protocol definitions
/usr/include/sys/xt.h	driver specific definitions

SEE ALSO

jagent(5), layers(5), termio(7), tty(7).
ioctl(2), open(2), libwindows(3X), in the *Programmer's Reference Manual*.
layers(1) in the *User's Reference Manual*.



NAME

intro – introduction to system maintenance procedures

DESCRIPTION

This section outlines certain procedures that will be of interest to those charged with the task of system maintenance. Included are discussions of such topics as boot procedures, recovery from crashes, file backups, etc.

SEE ALSO

System Administrator's Guide.



NAME

3B2boot – 3B2 computer bootstrap procedures

DESCRIPTION

The bootstrap procedure on the 3B2 computer consists of three phases. The first phase is initiated by the user typing `boot` while in firmware. This phase will prompt for the name of the file to be booted, then the device from which to boot. (This phase is also initiated after power-up, and upon completion of diagnostics. In this case, the file to boot defaults to `/unix` and the device to boot defaults to the first integral hard disk.) The second phase is the execution of `mboot` which reads from the boot block of the selected boot device. `mboot` is loaded at physical address 0x2004000. Its function is to read the `lboot` phase from the remainder of the boot partition into the top of the first half megabyte of main memory and to initiate `lboot` execution.

Any error during the `mboot` phase will cause a return to the firmware.

The `lboot` phase has been passed the name of the file to be booted. If the name given is a directory, `lboot` will respond with a listing of the files present in that directory. Once a valid file name has been obtained, `lboot` loads the file into memory and starts the execution of the file.

SEE ALSO

`newboot(1M)`, `a.out(4)`.

DIAGNOSTICS

Self-explanatory messages about bad directory entries and bad file formats.

BUGS

`lboot` isn't smart enough to know which `a.out` files can be used as bootable programs. If an `a.out` is specified that is not bootable, `lboot` will load it and branch to it. What happens after that is unpredictable. Note that if `/etc/system` is specified a self-configuration reboot will occur.



NAME

baud – display or change console baud rate

SYNOPSIS

baud
q

DESCRIPTION

baud is a firmware command that displays the console baud rate, then prompts for a new baud rate. It is not available in all ROM code issues. The default entry, <CR> will not change the console baud rate, otherwise the system will change the console's baud rate to the requested value and save the value in NVRAM, provided it is a supported rate. *baud* complains about unsupported baud rates. *baud* is available in ROM code with an issue number greater than or equal to 0x0c (12).

Q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).

SEE ALSO

version(8).



NAME

dgmon – run diagnostic phases in firmware

SYNOPSIS

```

dgn
dgn [unit[=number]]
dgn [unit[=number]] [ph=a[-b]] [rep=n] [ucl] [soak]
l(ist) unit
h(elp)
s(how)

```

(phase range specified ph=a-b means phases a through b)

DESCRIPTION

The Diagnostic Monitor Utility, *dgmon*, allows diagnostics to be run on the 3B2 Computer in the firmware mode via the system console. The particular diagnostic phases are specified via the **dg**n command. The interface and the entry into the diagnostic mode is discussed in detail in the *System Administrator's Guide*. Diagnostics can be invoked for the entire computer, types of devices (e.g., all ports boards), a specific device (e.g., the system board), or a particular phase or range of phases for the device or device type. Each diagnostic phase and phase description can be found by listing the diagnostic phases for each computer device (such as 1 sbd). The Monitor will list the 3B2 Computer physical devices with the s(how) command.

Types of diagnostic phases:

Normal-Diagnostics that run every time the computer is powered up

Demand-Diagnostics that run during soak or must be specifically requested

Interactive-Diagnostics that must be specifically requested and may cause loss of stored data or require operator intervention.

(Diagnostic Monitor will deny request for diagnostics on unequipped devices and non-existent phases)

Option definitions:

ucl -unconditional execution, run all specified phases and display all failing results

rep=n -repeat phases(s) n times

ph=a -select individual phase

ph=a-b -select phase range a through b

soak -silently and continuously run normal and demand diagnostics for specified range (default: all of phase table) and for specified repetitions (default: continuous-stopped with keyboard entry)

EXAMPLES

- ll. dgn (full system)
- dgn ports (all ports devices)
- dgn sbd 0 ucl (Unconditional execution)
- dgn ports 0 (ports 0 diagnostic)
- dgn ports 1 ucl
- dgn sbd ph=3 (phase specification)
- dgn sbd ph=1-5 (phase range specification)
- dgn sbd soak (diagnostic system board soak)

Whenever specific phases are requested, the device to be tested must be designated.

DIAGNOSTICS

Improperly typed syntax will generate message or specify the invalid input. The "h" command will generate a listing of the correct syntax for the system board firmware.

NAME

`edt` – Equipped Device Table commands

SYNOPSIS

```
edt  
q
```

DESCRIPTION

`edt` is a firmware command that displays the fields of the Equipped Device Table (EDT) entries, device by device. `edt` is useful when a check of the exact contents of the EDT will resolve uncertainties.

`q(uit)` provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).

SEE ALSO

`filledt(8)`.



NAME

filledt – fill Equipped Device Table (EDT)

SYNOPSIS

filledt

DESCRIPTION

The *filledt* command is a firmware-level routine that completes the Equipped Device Table (EDT) for the computer. It compares ID codes gathered for devices and subdevices to those in the look-up tables stored in */dgn/edt_data*. When ID matches between the EDT and the look-up table occur, *filledt* copies data, such as device names, into the EDT. The *filledt* command also makes a console terminal check.

The *filledt* command has two operating modes: power-up (automatic) and manual.

During the power-up sequence it silently completes the EDT and checks the console. An error message, **EDT FILL FAILED**, appears only if a peripheral device fails to respond to subdevice equipage queries.

In the manual request mode *filledt* completes the EDT, reporting success or failure. It checks for a console terminal and reports the console location and *c_flags* values.

FILES

/filledt

SEE ALSO

termio(7).



NAME

`mk` – remake the binary system and commands from source code

DESCRIPTION

All source code for the UNIX system is distributed in the directory `/usr/src`. The directory tree rooted at `/usr/src` includes source code for the operating system, libraries, commands, miscellaneous data files necessary for the system and procedures to transform this source code into an executable system.

Within the `/usr/src` directory are the `cmd`, `lib`, `uts`, `head`, and `stand` directories, as well as commands to remake the parts of the system found under each of these sub-directories. These commands are named `:mk` and `:mkdir` where `dir` is the name of the directory to be recreated. Each of these `:mkdir` commands will rebuild all or part of the directory it is responsible for. The `:mk` command will run each of the other commands in order and thus, recreate the whole system. The `:mk` command is distributed only to source code licensees.

Each command, with its associated directory, is described below.

`:mklib` The `lib` directory contains the source code for the system libraries. The most important of these is the C library. Each library is in its own sub-directory. If any arguments are specified on the `:mklib` command line then only the given libraries will be rebuilt. The argument `*` will cause it to rebuild all libraries found under the `lib` directory.

`:mkhead`

The `head` directory contains the source code versions of the header files found in the `/usr/include` directory. The `:mkhead` command will install the header files given as arguments. The argument `*` will cause it to install all header files.

`:mkuts` The `uts` directory contains the source code for the UNIX Operating System. The `:mkuts` command takes no arguments and invokes a series of makefiles that will recreate the operating system.

Associated with the operating system is a set of header files that describe the user interface to the operating system. The source for these header files is found in a sub-directory within the `uts` directory tree. The user-accessible versions of these header files are found in the `/usr/include/sys` directory. The `:mksyshead` command will install these header files into the `/usr/include/sys` directory.

`:mkstand`

The `stand` directory contains stand-alone commands and boot programs. The `:mkstand` command will rebuild and install these programs. Note that these stand-alone programs are only applicable to the DEC processors and will not be built for any other machine.

`:mkcmd` The `cmd` directory contains the source code for all the commands available on the system. There are two types of entries within the `cmd` directory: commands whose source code consists of only one file with one of the following suffixes: `.l`, `.y`, `.c`, `.s`, `.sh`, or a sub-directory that contains the multiple source files that comprise a particular

command or subsystem. Each sub-directory is assumed to have a makefile (see *make(1)*) with the name *command.mk* that will take care of creating everything associated with that directory and its sub-directories.

The *:mkcmd* command transforms source code into an executable command based upon a set of predefined rules. If the *:mkcmd* command encounters a sub-directory within the *cmd* directory then it will run the makefile found in that sub-directory. If no makefile is found then an error will be reported. For single file commands, the predefined rules are dependent on the file's suffix. C programs (*.c*) are compiled by the C compiler and loaded stripped with shared text. Assembly language programs (*.s*) are assembled and loaded stripped. Yacc programs (*.y*) and lex programs (*.l*) are processed by *yacc(1)* and *lex(1)* respectively, before C compilation. Shell programs (*.sh*) are copied to create the command. Each of these operations leaves a command in the *.cmd* directory which is then installed into a user-accessible directory by using */etc/install*.

The arguments to *:mkcmd* are either command names or subsystem names. The subsystems distributed with the UNIX system are: *acct*, *graf*, *sgs*, *sccs*, and *text*. Prefacing the *:mkcmd* command with an assignment to the shell variable *\$ARGS* will cause the indicated components of the subsystem to be rebuilt.

For example, the entire *sccs* subsystem can be rebuilt by:

```
/usr/src/:mkcmd sccs
```

while the *delta* component of *sccs* can be rebuilt by:

```
ARGS="delta" /usr/src/:mkcmd sccs
```

The *log* command, which is a part of the *stat* package, which is itself a part of the *graf* package, can be rebuilt by:

```
ARGS="stat log" /usr/src/:mkcmd graf
```

The argument *** will cause all commands and subsystems to be rebuilt.

Makefiles throughout the system, and particularly in the *cmd* directory, have a standard format. In particular, *:mkcmd* depends on each makefile having target entries for *install* and *clobber*. The *install* target should cause everything over which the makefile has jurisdiction to be built and installed by */etc/install*. The *clobber* target should cause a complete cleanup of all unnecessary files resulting from the previous invocation.

An effort has been made to separate the creation of a command from source and its installation on the running system. The command */etc/install* is used

by `:mkcmd` and most makefiles to install commands in standard directories on the system. The use of `install` allows maximum flexibility in the administration of the system. The `install` command makes very few assumptions about where a command is located, who owns it, and what modes are in effect. All assumptions may be overridden on invocation of the command, or more permanently by redefining a few variables in `install`. The purpose of `install` is to install a new version of a command in the same place, with the same attributes as the prior version.

In addition, the use of a separate command to perform installation allows for the creation of test systems in other than standard places, easy movement of commands to balance load, and independent maintenance of makefiles.

SEE ALSO

`install(1M)`.

`lex(1)`, `make(1)`, `yacc(1)` in the *Programmer's Reference Manual*.



NAME

newkey – makes a floppy key for the 3B2 Computer

SYNOPSIS

newkey
q

DESCRIPTION

newkey is a firmware command that uses a formatted floppy disk to store the key that clears the Non-Volatile RAM (NVRAM) during the power-up or system reset sequences. The floppy disk on which the key resides is called the 'floppy key'. It is often used to reset the firmware passwd to its default, *mcp*.

Q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).

EXAMPLE

Enter name of program to execute []: **newkey**

Creating a floppy key to enable clearing of the saved NVRAM information

Insert a formatted floppy, then type 'go' (q to quit): **go**

Creation of floppy key complete



NAME

`passwd` – change firmware password

SYNOPSIS

```
passwd  
q
```

DESCRIPTION

The *passwd* firmware command re-assigns the firmware password for a 3B2 computer. It is similar to the system-level command. A firmware password may not exceed 8 characters.

EXAMPLE

```
Enter name of program to execute [ ]: passwd  
enter old passwd: <old passwd>  
enter new passwd: <new passwd>  
confirmation: <new passwd>
```

SEE ALSO

`passwd(1)` in the *User's Reference Manual*.



NAME

ports – create character device files and inittab entries for ports boards

SYNOPSIS

/etc/ports

DESCRIPTION

/Etc/ports will create character device files in */dev* and add new entries in */etc/inittab* for 4 asynchronous RS-232 ports and 1 parallel printer port. Each port will be named

tty[slot#] [1-5]

If the board configuration has changed, *ports* will do the following:

Remove any tty device files for a board that no longer resides in a slot.

Remove device files of other boards such as the NI if a ports board now resides in the slot that previously held an NI. A warning message would be sent to the console that a device file was being removed.

Make new tty device files for the ports boards if needed.

Make new inittab entries for the ports boards.

If the configuration has not changed, the *ports* program will exit without doing anything.

Any devices, such as a printer or a modem, that are added to a ports board should link the names that are to be used for the devices to the corresponding tty device files that were created (see *ln*[1]). The command can be used only by the super-user.

EXAMPLE

A parallel printer is added to a ports board that is in slot 1. The corresponding slot would be tty15. The user should use *ln* to link an appropriate name such as lp1 to the tty device file.

ln /dev/tty15 /dev/lp1

FILES

/etc/inittab
/dev/tty[slot#][1-5]

SEE ALSO

ln(1) in the *User's Reference Manual*.

WARNINGS

A warning will be issued for each device file that is removed provided it is not a tty device file. If a ports board has been removed and lp1 has been linked to a tty device file, the message would be as follows:

Warning: /dev/lp1 is being removed.



NAME

`sysdump` – boot option to dump system memory image to floppy disk(s)

SYNOPSIS

`sysdump`

DESCRIPTION

The `sysdump` command dumps the system memory image to one or more floppy disks depending on the size of memory and user request. This memory image can later be analyzed by `crash(1M)`. `sysdump` is invoked as a boot option.

When booted, `sysdump` begins an interactive procedure that prompts the user to insert the floppies to be loaded. The user has the option of quitting the session any time. This allows only the portion of the system image needed to be dumped. floppies.

The output of `sysdump` provides one input to `crash(1)`. The other input is the text file that was used to boot this system image. This is needed to provide symbolic reference to the system dump. The text file must be manually saved after the machine has been booted. If `/unix` was booted then this should be dumped to floppy to accompany the system dump.

FILES

`/dev/c0d0s6` -- device used for floppy access

`/unix` -- the text file typically used to boot the machine

SEE ALSO

`crash(1M)`, `ldsysdump(1M)`.

DIAGNOSTICS

If a floppy diskette is inserted out of sequence, a message is printed. The user is allowed to insert a new diskette and continue the session.

WARNINGS

It is critical to provide access to the text file used to boot the machine. This file must be saved.

The diskettes should be labeled clearly so they can be loaded in the proper sequence.



NAME

version – version commands

SYNOPSIS

version

q

DESCRIPTION

The *version* firmware command lists information about the ROM code in the 3B2 Computer system board. It prints data about the ROM issue, date, software load and serial number.

Q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).





What do YOU think?

AT&T values your opinion. Please indicate your opinions in each of the following areas. We'd like to know how well this document meets your needs.

Book Title: _____

	Excellent	Good	Fair	Poor
Accuracy – Is the information correct?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness – Is information missing?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization – Is information easy to find?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity – Do you understand the information?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples – Are there enough?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Illustrations – Are there enough?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Appearance – Do you like the page format?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Physical binding – Do you like the cover and binding?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Does the document meet your needs? Why or why not?

What is the single most important improvement that we could make to this document?

Please complete the following information.

Name (Optional): _____

Job Title or Function: _____

Organization: _____

Address: _____

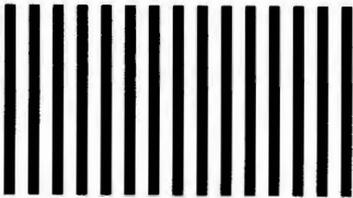
Phone (): _____

If we need more information may we contact you? Yes No

Thank you.



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 199 SUMMIT, NJ

POSTAGE WILL BE PAID BY ADDRESSEE

Department Head
Documentation and User
Interface Technologies Department
AT&T
Room F-308
190 River Road
Summit, NJ 07901-9907

