AT&T

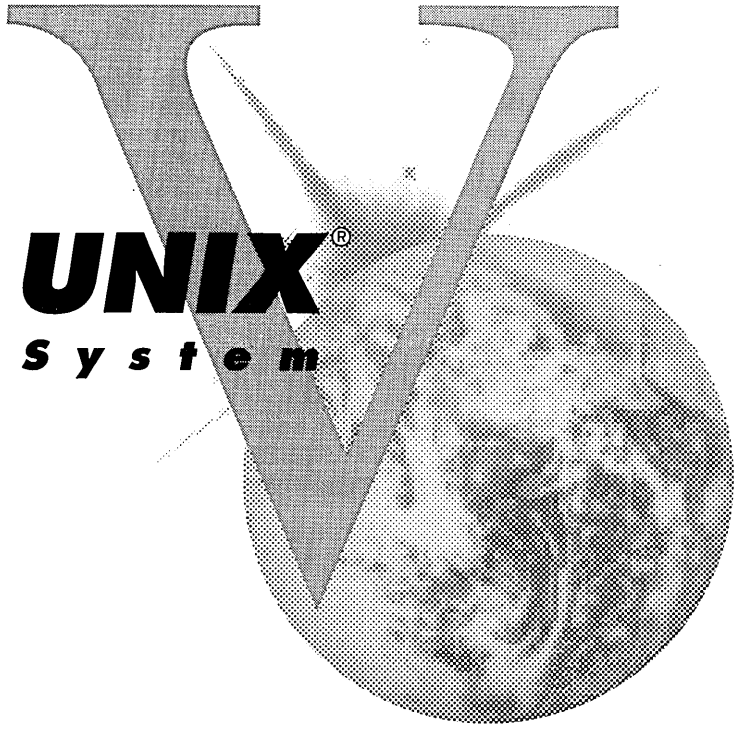
UNIX® SYSTEM V RELEASE 4

BSD/XENIX® Compatibility Guide

UNIX Software Operation

UNIX® SYSTEM V RELEASE 4

BSD/XENIX® Compatibility Guide



UNIX Software Operation

Published by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

IMPORTANT NOTE TO USERS

While every effort has been made to ensure the accuracy of all information in this document, AT&T assumes no liability to any party for any loss or damage caused by errors or omissions or by statements of any kind in this document, its updates, supplements, or special editions, whether such errors are omissions or statements resulting from negligence, accident, or any other cause. AT&T further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. AT&T disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, *including implied warranties of merchantability or fitness for a particular purpose.* AT&T makes no representation that the interconnection of products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting or license to make, use or sell equipment constructed in accordance with this description.

AT&T reserves the right to make changes without further notice to any products herein to improve reliability, function, or design.

TRADEMARKS

BBN is a registered trademark of Bolt Beranek and Newman, Inc.
Concept is a trademark of Human Designed Systems.
Datamedia is a registered trademark of Datamedia Corporation.
DEC is a registered trademark of Digital Equipment Corporation.
Diablo is a registered trademark of Xerox Corporation.
GSI is a trademark of General Systems International.
Hewlett-Packard is a registered trademark of Hewlett-Packard Co.
Imagen is a registered trademark of Imagen Corporation.
PostScript is a registered trademark of Adobe Systems, Inc.
Tektronix is a registered trademark of Tektronix, Inc.
UNIX is a registered trademark of AT&T
Versatec is a registered trademark of Versatec.
Wyse is a registered trademark of Wyse Technologies.
XENIX is a registered trademark of Microsoft, Inc.
Xylogics is a registered trademark of Xylogics, Inc.

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-933664-8

UNIX
PRESS
A Prentice Hall Title

P R E N T I C E H A L L

ORDERING INFORMATION

UNIX® SYSTEM V, RELEASE 4 DOCUMENTATION

To order single copies of UNIX® SYSTEM V, Release 4 documentation, please call (201) 767-5937.

ATTENTION DOCUMENTATION MANAGERS AND TRAINING DIRECTORS:
For bulk purchases in excess of 30 copies please write to:
Corporate Sales
Prentice Hall
Englewood Cliffs, N.J. 07632.
Or call: (201) 592-2498.

ATTENTION GOVERNMENT CUSTOMERS: For GSA and other pricing information please call (201) 767-5994.

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

AT&T UNIX® System V Release 4

General Use and System Administration

UNIX® System V Release 4 Network User's and Administrator's Guide
UNIX® System V Release 4 Product Overview and Master Index
UNIX® System V Release 4 System Administrator's Guide
UNIX® System V Release 4 System Administrator's Reference Manual
UNIX® System V Release 4 User's Guide
UNIX® System V Release 4 User's Reference Manual

General Programmer's Series

UNIX® System V Release 4 Programmer's Guide: ANSI C and Programming Support Tools
UNIX® System V Release 4 Programmer's Guide: Character User Interface (FMLI and ETI)
UNIX® System V Release 4 Programmer's Guide: Networking Interfaces
UNIX® System V Release 4 Programmer's Guide: POSIX Conformance
UNIX® System V Release 4 Programmer's Guide: System Services and Application Packaging Tools
UNIX® System V Release 4 Programmer's Reference Manual

System Programmer's Series

UNIX® System V Release 4 ANSI C Transition Guide
UNIX® System V Release 4 BSD / XENIX® Compatibility Guide
UNIX® System V Release 4 Device Driver Interface / Driver–Kernel Interface (DDI / DKI) Reference Manual
UNIX® System V Release 4 Migration Guide
UNIX® System V Release 4 Programmer's Guide: STREAMS

Contents

INTRODUCTION

INTRODUCTION

Introduction

Two major variants of the UNIX Operating System have been merged into System V Release 4. Because of certain conflicts, not all commands of each version of the UNIX System have been merged into the System V base. However, to make the transition as easy as possible, many of these commands have been retained as part of a Compatibility Package.

How This Guide Is Organized

There are two sections in this book. For each of the UNIX variants that have been merged into System V Release 4, this book contains a discussion of the commands, system calls and library routines that were not merged into the base.

Organization of the Sections

Each section in this book contains three parts:

1. overview
2. manual pages
3. subject index

The overview gives a high level description and, in some cases, a tutorial for many of the commands and routines included in the Compatibility Package. It also describes steps you must take to use these commands.

Following the overview, you will find a table of contents and permuted index, and manual pages describing each of the commands, system calls and library routines in the package.

Finally, you'll find a subject index for both the Berkeley and XENIX sections. Each section is fully and independently indexed. For a comprehensive index, see the *Product Overview and Master Index.*

Notation Conventions Used in This Guide

This section describes the notation conventions used in this book.

Computer Input and Output

- References to literal computer input and output (such as commands entered by the user or screen messages produced by the system) are shown in a monospace font, as in the following example:

```
$ ls -l report.oct17
-rw-r--r--   1 jim     doc       3239 May 26 11:21 report.oct17
```

- Substitutable text elements (that is, text elements that you are expected to replace with specific values) are shown in an italic font, as in the following example:

  `$ cat` *filename*

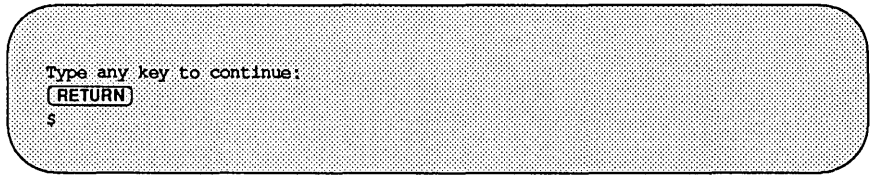
  The italic font is a signal that you are expected to replace the word *filename* with the name of a file.

- Comments in a screen display–that is, text that is not computer output, but is an aside from the author to the reader–is shown in an *italic* font and is indented, as in the following example:

```
                .
                .
                .
        command interaction
                .
                .
                .
                .
Press RETURN to continue.
```

- Instructions to the reader to type input usually do not include explicit instructions to press the RETURN key at the appropriate times (such as after entering a command or a menu choice) because this instruction is implied for all UNIX system commands and menus.

  In one circumstance, however, an instruction to press the RETURN key is explicitly provided: when, during an interactive routine, you are expected to press RETURN without having typed any text, an instruction to do so will be provided, as follows:



- Control characters are shown by the string CTRL-*char* where *char* is a character such as "d" in the control character CTRL-d. To enter a control character, hold down the CTRL key and press the letter shown. Be sure to type the letter exactly as specified: when a lower case letter is shown (such as the "d" in the example above), enter that lower case letter. If a character is shown in upper case (such as CTRL-D), you should enter an upper case letter.
- The system prompt signs shown in examples of interactive sessions are the standard default prompt signs for AT&T UNIX System V Release 4.0:
  - the dollar sign (`$`) for an ordinary user
  - the pound sign (`#`) for the owner of the `root` login

Admonishments

An admonishment is a short piece of text that is set off from the main body of the text and is marked with an icon to show that it deserves special attention. The only type of admonishment used in this guide is a note.

Text marked with a NOTE icon is material that emphasizes points of interest, presents extended parenthetical information, or cites references to other documents and software. Information contained in a note may help you avoid inconvenience (rather than injury or damage to the system). The following is an example of a NOTE.

To start and stop the LP print service manually, you must be logged in as either `root` or `lp`.

Related Documents and Training Courses

For a complete list of books about AT&T UNIX System V Release 4.0, see the *Product Overview and Master Index* for this release. To order books, call one of the following numbers:

In the continental U.S.: 1-800-432-6600

Outside the continental U.S.: 1-800-256-1242

Outside the U.S.: 317-256-1242

For a complete list of training courses related to AT&T UNIX System V Release 4.0, see one of the following:

AT&T Education and Training Catalog of Courses

AT&T UNIX System V and C Language Training
(brochure)

For information about taking a course, call one of the following numbers:

In the continental U.S.: 1-800-554-6400

Outside the continental U.S.: 201-658-6747

How to Comment on This Guide

We want to provide you with the best possible documents for UNIX System V Release 4.0. If you have suggestions about how we can improve this book, please send them to us by filling out and mailing the card located after the title page.

Contents

Overview

UNIX System V Release 4.0 is, among other changes, a merge of UNIX System V with the enhanced BSD UNIX Operating System. (BSD UNIX is also known as "Berkeley UNIX.") In most cases these changes have been implemented to preserve compatibility with System V.

However, Release 4.0 includes a "compatibility package" that provides a full BSD environment. It contains the following, which are either different from UNIX System V or don't exist in UNIX System V Release 4.0:

- **BSD commands.** For example, BSD's `look` command is not in UNIX System V. Another example is the `df` command, which reports block sizes differently in BSD than in UNIX System V. A third example is the `ln` command; it has a `-f` option that has different meanings in the two environments.
- **Library routines.** For example, `printf` returns different values under BSD and UNIX System V; the BSD routine `re_comp` doesn't exist in UNIX System V.
- **Header files.** For example, `sysexits.h` doesn't exist in UNIX System V; and the BSD `sys/file.h` has additional `#defines` not in UNIX System V.
- **System Calls and Signals.** Routines to emulate BSD system calls. For example, the BSD system call, `killpg`, has been implemented as a library routine with the same name.

By using the compatibility package, you can take advantage of full BSD environment compatibility.

Why Use the Compatibility Package

Use the compatibility package if you:

- want BSD behavior of commands and routines.
- are more familiar with BSD and want a BSD environment.
- are using programs or shell scripts that make use of BSD commands, libraries, and routines.

Installing the Compatibility Package

The compatibility package is an optional software package. To install the compatibility package, see the *Source Code Provision Build Instructions*.

Accessing the Compatibility Package

Once the compatibility package is installed, you can make use of it by setting your `PATH` variable so that `/usr/ucb` comes before the default UNIX System V path directories `/usr/bin` and so on. `/usr/ucb` contains the compatibility package commands.

To find out what your path is currently, use the `set` command:

```
$ set
EDITOR=/usr/ucb/vi
HOME=/home/medici
LOGNAME=medici
PATH=.:/home/medici/bin:/usr/bin:/usr/ucb:/etc
PWD=/home/medici/att/compat
SHELL=/bin/sh
USER=medici
home=/home/medici
$
```

(Actually, you'll probably get a lot more information than this example shows.) To make the compatibility package commands the default, switch the order of `/usr/bin` and `/usr/ucb`:

```
$ PATH=.:/home/medici/bin:/usr/ucb:/usr/bin:/etc
$
```

To make this change permanent, put the above line in your `.profile` file (in your home directory).

Or your `.cshrc` file if you're using the C shell.

To avoid seeing all the environment variables you have set, you can use `echo`.

```
$ echo $PATH
PATH=.:/home/medici/bin:/usr/bin:/usr/ucb:/etc
$
```

Compatibility package header files and libraries called by the C compiler (`cc`) and linker (`ld`) are located in `/usr/ucbinclude` and `/usr/ucblib`.

By setting your path as shown above, you'll use `/usr/ucb/cc` when you compile C programs. `/usr/ucb/cc` sets its default paths to pick up, in this order:

1. User-specified include directories and libraries;
2. The compatibility include files and directories;
3. The default UNIX System V headers and libraries, if unresolved symbols remain.

Selected Features of the Compatibility Package

Printing Commands

The compatibility package contains five BSD printing commands that are not in the basic UNIX System V: `lpr`, `lpq`, `lprm`, `lpc`, and `lptest`. This is how they compare with UNIX System V commands:

UNIX System V Equivalents of BSD Printing Commands	
BSD Command	*UNIX System V Equivalent*
`lpr`	`lp`
`lpq`	`lpstat`
`lprm`	`cancel`
`lpc`	*no equivalent*
`lptest`	*no equivalent*

Each of these BSD commands contained in the compatibility package is explained further in Chapter *brief_disc*. Here is an overview of some important features:

1. There are no BSD commands for administration, maintenance, or accounting of printers; you must use UNIX System V schemes. The exception is the `lpc` command, which is included.
2. The BSD `printcap` file does not exist. Instead, use `/usr/share/lib/terminfo`.
3. Printing spooling commands do not depend on the `lpd` daemon, as in BSD. Instead, they use an HPI (High Performance Interface) with the UNIX System V printing daemon, `lpsched`.

For more information on each of these commands, see Chapter *brief_disc*.

Text Formatting and Bibliography Commands

The compatibility package includes a complete set of text formatting and bibliography commands. They are explained further in the manuals TEXT and TROFF. Additional information is provided in the Reference section of this document.

Text Formatting and Bibliography in the Compatibility Package	
`checkeq`	checks `eqn` source
`checknr`	checks `troff` and `nroff` source
`deroff`	removes `troff` and `nroff` codes
`diffmk`	compares `troff/nroff` files
`eqn`	formats equations
`eqnchar`	special characters for `eqn`
`neqn`	`eqn` for `nroff`
`nroff`	formatter for typewriter-like printers
`refer`	bibliography system
`soelim`	resolves `.so` calls
`tbl`	formats tables
`troff`	device-independent text formatter for phototypesetters
`me,ms,man`	`troff/nroff` macro packages

Library Routines and Header Files

The Compatibility Package also includes a set of compatibility libraries. These libraries consist of

a. routines not in UNIX System V, or

b. having a different interface and a different system call "wrapper" than their UNIX System V counterparts.

In the case of header files, differences remain, and the user should make himself or herself aware of them; however, an attempt has been made to achieve real source compatibility.

Mail Commands

The BSD version of `mail`, based on `sendmail`, has been included; it has a different user interface than UNIX System V's `mail`. Additionally, several useful mail facilities are included for compatibility. You should not mix and match commands from the two versions of the mail feature.

Mail Commands in the Compatibility Package	
`biff`	immediate notification of mail
`mailstats`	provides statistics on mail
`newaliases`	rebuilds alias tables
`sendmail`	basic mail utility
`vacation`	automatically replies to mail

BSD Compatibility Package Contents

Commands

The following BSD commands are not in UNIX System V Release 4.0 but are part of the Compatibility Package:

Commands Not in UNIX System V:

addbib	logger	newfs	syslogd
apropos	look	nroff	tbl
arch	lookbib	pagesize	tcopy
biff	lpc	printenv	troff
catman	lpq	reboot	tset
checknr	lpr	refer	ul
diffmk	lprm	renice	unifdef
fastboot	lptest	reset	uptime
fsirand	mach	roffbib	users
halt	mailstats	sccs	w
hostid	man	sendmail	whatis
hostname	mt	soelim	which
indxbib	newaliases	sortbib	whoami
lastcomm			

The following compatibility package commands are different from existing UNIX System V ones:

Commands Varying From UNIX System V

basename	echo	ls	shutdown
cc	eqn	neqn	stty
checkeq	groups	plot	sum
chown	grpck	prt	test
deroff	install	ps	tr
df	ld	pwck	vacation
du	ln		

Library Routines

The following compatibility package library routines do not exist in UNIX System V:

Library Routines Not in UNIX System V

_longjmp	firstkey	mult
_setjmp	floatingpoint	nextkey
alloca	fp_class	openlog
alphasort	ftime	pow
bcmp	gcd	random
bcopy	gconvert	re_comp
bzero	getdtablesize	re_exec
closelog	gethostid	reboot
copysign	gethostname	rindex
dbm_clearerr	getpagesize	rpow
dbm_close	getpriority	scalbn
dbm_delete	getrusage	scandir
dbm_error	getusershell	sdiv
dbm_fetch	getwd	seconvert
dbm_firstkey	ieee_functions	setbuffer
dbm_nextkey	ieee_handler	sethostname
dbm_open	index	setlinebuf
dbm_store	initstate	setlogmask
dbmclose	isnan	setpriority
dbminit	itom	setregid
decimal_to_double	killpg	setreuid
decimal_to_extended	madd	setstate
decimal_to_single	mcmp	setusershell
delete	mctl	sfconvert
double_to_decimal	mdiv	sgconvert
econvert	mfree	sigblock
endusershell	min	sigfpe
extended_to_decimal	mkstemp	siginterrupt
fconvert	mout	sigmask
fdopen	msqrt	sigpause
fetch	msub	sigsetmask
ffs	mtox	sigstack

Library Routines Not in UNIX System V (Continued)

sigvec	sys_siglist	utimes
single_to_decimal	syscall	wait3
srandom	syslog	wiFEXITED
store	timezone	wiFSIGNALED
strcasecmp	ualarm	wiFSTOPPED
strncasecmp	usleep	xtom

The following compatibility package library routines are different from those in UNIX System V Release 4.0:

Library Routines Varying from UNIX System V

fopen	printf	siglongjmp	stejmp
fprintf	psignal	signal	times
freopen	rand	sigsetjmp	vfprintf
gettimeofday	regex	sleep	vprintf
longjmp	setbuf	sprintf	vsprintf
nice	settimeofday	srand	wait
nlist	setvbuf		

Manual Pages Describing Library Routines	
Manual Page	**Routines**
alloca	alloca
bstring	bcopy, bcmp, bzero, ffs
dbm	dbminit, dbmclose, delete, fetch, firstkey, store, nextkey
decimal_to_floating	decimal_to_single, decimal_to_double, decimal_to_extended
econvert	econvert, fconvert, gconvert, seconvert, sfconvert, sgconvert
floating_to_decimal	single_to_decimal, double_to_decimal, extended_to_decimal
floatingpoint	floatingpoint
fopen	fopen, freopen, fdopen
ftime	ftime

Manual Pages Describing Library Routines	
Manual Page	**Routines**
getdtablesize	getdtablesize
gethostid	gethostid
gethostname	gethostname, sethostname
getpagesize	getpagesize
getpriority	getpriority, setpriority
getrusage	getrusage
gettimeofday	gettimeofday, settimeofday
getusershell	getusershell, setusershell, endusershell
getwd	getwd
ieee_functions	fp_class, isnan, copysign, scalbn
ieee_handler	ieee_handler
index	index, rindex
killpg	killpg
mctl	mctl
mkstemp	mkstemp
mp	madd, msub, mult, mdiv, mcmp, min, mout, pow, gcd, rpow, msqrt, sdiv, itom, xtom, mtox, mfree
ndbm	dbm_open, dbm_close, dbm_fetch, dbm_store, dbm_delete, dbm_firstkey, dbm_nextkey, dbm_error, dbm_clearerr
nice	nice
nlist	nlist
printf	printf, fprintf, sprintf, vprintf, vfprintf, vsprintf
psignal	psignal, sys_siglist
rand	rand, srand
random	initstate, setstate, random, srandom
reboot	reboot

Manual Pages Describing Library Routines	
Manual Page	Routines
regex	regex, re_comp, re_exec
scandir	scandir, alphasort
setbuf	setbuf, setbuffer, setlinebuf, setvbuf
setbuffer	setbuffer, setlinebuf
setjmp	setjmp, longjmp, _setjmp, _longjmp, sigsetjmp, siglongjmp
setregid	setregid
setreuid	setreuid
sigblock	sigblock, sigmask
sigfpe	sigfpe
siginterrupt	siginterrupt
signal	signal
sigpause	sigpause
sigsetmask	sigsetmask
sigstack	sigstack
sigvec	sigvec
sleep	sleep
string	strcasecmp, strncasecmp
syscall	syscall
syslog	syslog, openlog, closelog, setlogmask
times	times
timezone	timezone
ualarm	ualarm, usleep
usleep	usleep
utimes	utimes
wait	wait, wait3, WIFSTOPPED, WIFSIGNALED, WIFEXITED

Signals

The following signals are in the compatibility package:

Signals

_longjmp	longjmp	sigblock	sigpause
_psignal	psignal	siginterrupt	sigsetmask
_setjmp	setjmp	signal	sigvec

System Calls

The following system calls are in the compatibility package:

System Calls

getdtablesize	gettimeofday	setreuid	wait3
gethostid	killpg	settimeofday	
gethostname	reboot	syscall	
getrusage	setregid	utimes	

Header Files

The following compatibility package header files do not exist in UNIX System V Release 4.0:

Header Files Not in Release 4.0

dbm.h	strings.h	sys/mtio.h
fp.h	struct.h	sys/reboot.h
mp.h	sunfp.h	sysexits.h
ndbm.h	sys/ieeefp.h	ufs/quota.h

The following compatibility package header files differ from the header files found in UNIX System V Release 4.0:

Header Files Varying from UNIX System V

assert.h	sys/dirent.h	sys/signal.h
regexp.h	sys/fcntl.h	sys/types.h
setjmp.h	sys/file.h	sys/vfs.h
signal.h	sys/param.h	sys/wait.h
stdio.h	sys/resource.h	unistd.h

BSD Compatibility Package Commands

Data Manipulation

`echo` The BSD and UNIX System V versions of `echo` differ in their handling of the `-n` option, and in the recognition of certain escape sequences.

The built-in version of `echo` has been modified to have BSD behavior if `/usr/ucb` is placed before `/usr/bin` in the `PATH` environment variable. Since `/usr/bin/echo` has not been modified, a BSD-compatible version of `echo` has been placed in the compatibility package.

`look` Looks up a given word or string in the system dictionary. Not in UNIX System V.

`tr` translates characters from the standard input to the standard output. There are two differences between the UNIX System V version and the BSD version:

- In BSD, when translating from *string1* to *string2*, if *string2* is shorter than *string1*, it (*string2*) is padded out to the length of *string1* by repeating *string2*'s last character.
- The BSD version does not require enclosing alphanumeric ranges inside square brackets.

Display

`catman` Creates the preformatted versions of the on-line Manual Pages from their `nroff` source. Not in UNIX System V.

`man` Displays the selected Manual Page. Not in UNIX System V.

`plot` Graphics filters for various plotters. This version has more filters than the UNIX System V version, allowing you to run it on more kinds of terminals.

`reset` and `tset`
: These commands are similar to `tput reset` and `tput init`, respectively; that is, they set and reset terminal behavior characteristics. However they differ enough to be included here. `reset` and `tset` are not in UNIX System V.

Text Processing and Bibliography

The following commands are all used for formatting text or generating a bibliography. They are described in greater detail in the manuals **TEXT** and **TROFF**. (Manual Pages for these commands are also included at the end of this document.)

None of the text processing commands is included in UNIX System V's base.

`checkeq`
: Checks that your `eqn` (described below) input is correct.

`checknr`
: Checks your `nroff` and `troff` input for errors.

`deroff`
: Removes `troff` and `nroff` code from documents.

`diffmk`
: Compares two `nroff` or `troff` source files and makes a third file containing "change mark" (`.mc`) notations showing where the two files differ.

`eqn`
: This is a preprocessor for typesetting mathematical equations. Used with `troff`.

`eqnchar`
: A set of special characters for `eqn`.

`ms`, `me`, and `man`
: These are macro packages for use with `troff` and `nroff`. `man` is used for formatting Manual Pages. `me` and `ms` are used for formatting technical and general manuscripts, respectively.

`neqn`
: Similar to `eqn`, but for use with `nroff` instead of `troff`.

`nroff`
: A type processing utility for formatting text to be printed on typewriter-like printers.

`refer` — A bibliography system supporting data entry, indexing, retrieval, sorting, and footnote or endnote numbering. It is a `nroff` and `troff` preprocessor (like `eqn` or `tbl`). These `refer` commands all relate to creating or maintaining a bibliography:

- `addbib` creates and extends the bibliographic database;
- `indxbib` creates an index to the references;
- `lookbib` quickly retrieves individual citations or groups of citations;
- `roffbib` runs off the entire database, formatting it not as footnotes, but as a bibliography or annotated bibliography;
- `sortbib` sorts the bibliography by author and date, or other criteria.

`soelim` — Eliminates `.so` requests from `troff` and `nroff` documents. `.so` is used to "source" other documents; that is, you use `.so` when you want the contents of another document read in at that point. However, in some cases using `.so` does not work (such as with `tbl`), and it's better to simply put the sourced document in. `soelim` replaces `.so` requests with the sourced document.

`tbl` — A text formatting utility for making tables. The tables in this document were created with `tbl`.

`troff` — A device-independent text formatter for use with phototypesetters. This document was formatted with `troff`.

`ul` — Designed for CRTs, this command highlights underlined text using a terminal's underline mode, if available, and otherwise reverse video mode.

File Management

`basename` Strips the full path name off a file. If you follow the path name with a suffix, that suffix is stripped, too:

```
$ basename /usr/src/rundog.c .c
rundog
$
```

`basename` is useful for writing shell scripts.

This version parses differently from the UNIX System V one.

`chown` The compatibility package version allows an optional *.group* suffix on the *user* argument, to allow the changing of both owner and group in a single operation.

`ln` Much BSD behavior has been incorporated into the Release 4.0 version of `ln`. Chiefly, `ln` in BSD doesn't remove an existing target, while the version in UNIX System V does, if it has the requisite permissions. Because the BSD behavior is not obtained by default, the compatibility package contains a version of `ln`, so that no special options are necessary.

`ls` The compatibility package version of `ls` differs from UNIX System V version in three ways:

- the `-s` option prints only the user name (not group);
- the `-s` option reports block sizes in blocks of 1024 bytes, instead of 512;
- this version of `ls -F` prints an equals sign (=) to indicate AF_UNIX address family sockets.

`sum` This version always calculates checksums in a machine-independent way. Also, it uses 1024-byte (not 512-byte) blocks.

`test` The BSD and UNIX System V versions of `test` differ with respect to the `-f` option. In BSD, this option means "is the target not-a-directory?" while in UNIX System V it means "is the target a plain file?"

As with `echo`, the builtin version of `test` has been modified to have BSD behavior when `/usr/ucb` is placed before `/usr/bin` in the `PATH` environment variable. Since `/usr/bin/test` has not been modified, a BSD-compatible version of `test` has been placed in the compatibility package.

`which` Tells you which version of a command you normally use:

```
$ which tr
/usr/bin/tr
$
```

There is no equivalent for this command in UNIX System V.

File System Management

`df` The most important difference is that block sizes are different from the UNIX System V version; also, this version preserves the `-t` option to mean "report on filesystems of a given type."

`du` Gives the number of disk blocks used by files in a directory. Useful for seeing how disk space is being used. This version uses 1024-byte block sizes, while the UNIX System V version uses 512-byte blocks.

Languages

`cc` Compatibility package header files and libraries called by the C compiler (`cc`) and linker `ld`) are located in `/usr/ucbinclude` and `/usr/ucblib`. The compatibility package version of `cc` looks in `/usr/ucbinclude` and `/usr/ucblib` to find BSD library and header files.

Actually, `cc` in the compatibility package, is a shell script that sets the paths of the include directories.

1. User-specified include directories and libraries;
2. The compatibility include files and directories;
3. The default UNIX System V headers and libraries, if unresolved symbols remain.

`ld` As with `cc`, a shell script in the compatibility package ensures that libraries in `/usr/ucblib` are linked in, instead of the default UNIX System V libraries.

When unresolved symbols remain, the libraries in `/usr/lib` are referenced. Note that `/usr/ucblib/libucb.a` is always linked when the shell script is invoked.

`prt` This SCCS command displays the "delta" and commentary record for an SCCS file. See `prs` because `prt` is obsolescent. Not in UNIX System V.

`sccs` The front-end program to the Source Code Control System (SCCS). SCCS allows you to restrict access to programs or documents that are being worked on by several people at once. Not in UNIX System V.

`unifdef` Strips `#ifdef` statements from C code. Understands ANSI C preprocessor statements. Not in UNIX System V.

Printing Commands

`lpc` — `lpc`, the line printer control program, is the same as in BSD except that it does not make use of the file `/etc/printcap`. Use `lpc` to start and stop a printer, disable or enable a spooling queue, rearrange the order of printing jobs, or display the status of each printer. Not in UNIX System V.

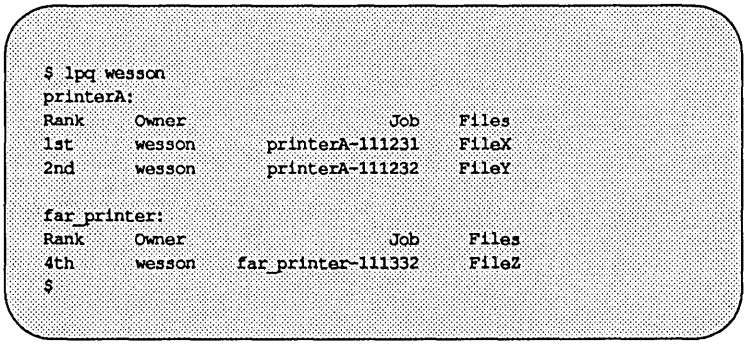
`lpq` — `lpq` is the equivalent of the UNIX System V command `lpstat`.

`lpq` returns information on all print jobs on the local printer; however, for remote printers, it provides the status only for jobs belonging to the user.

The command `lpq` *username* provides the status of all files, on all machines (both local and remote), belonging to the named user. In these examples, user `wesson` sends files `fileX` and `fileY` to the local printer, `printerA`, and `fileZ` to a remote printer:

```
$ lpr fileX
$ lpr fileY
$ lpq
Rank    Owner              Job        Files
active  smith        printerA-111230   file1
1st     wesson        printerA-111231   fileX
2nd     wesson        printerA-111232   fileY
3rd     jones         printerA-111233   file9
$
```

```
$ lpr -d far_printer fileZ
$ lpq -Pfar_printer
Rank    Owner                  Job     Files
4th     wesson    far_printer-111332   fileZ
$
```



```
$ lpq wesson
printerA:
Rank     Owner                Job    Files
1st      wesson       printerA-111231   FileX
2nd      wesson       printerA-111232   FileY

far_printer:
Rank     Owner                Job    Files
4th      wesson    far_printer-111332   FileZ
$
```

`lpr` — `lpr` is the BSD equivalent of `lp`, the command for sending files to a printer. The compatibility package version of `lpr` does not support the `-r` option (which removes a file after spooling it). Nor does it allow for specific fonts to be mounted on font positions 1, 2, 3, or 4.

`lprm` — This is the BSD equivalent of the UNIX System V `cancel` command. It removes files from the print queue.

`lptest` — This produces a ripple pattern for testing printers.

Mail

There are some differences between the mail commands in BSD and UNIX System V. If you are using the Compatibility Package's mail, you should also use all of its related programs (listed here) as well. You should not mix and match commands from the two versions of the mail feature.

`biff` — `biff` enables and disables immediate notification of the arrival of mail. Not in UNIX System V.

`mailstats` — Reports statistics gathered by `sendmail`, including number of messages received, number sent, and their size. Not in UNIX System V.

`newaliases` Rebuilds the aliases database in `/usr/ucblib/aliases`; called by `sendmail`, it's run every time a message is sent. Not in UNIX System V.

`sendmail` The basic mail utility for which `mail` is a user-friendly "front-end." `sendmail` has been modified for additional security. Not in UNIX System V.

`vacation` The `vacation` program sends a pre-written reply to anyone who sends you mail while you are gone.

NFS

`fsirand` Installs random inode generation numbers on all the inodes on a given device; also installs a filesystem ID in the superblock. This helps increase the security of filesystems exported by NFS. Not in UNIX System V.

UNIX System V Management

Because of limitations in underlying UNIX System V support, the commands in this section for stopping and starting your system cannot provide full BSD functionality. Nonetheless, they are included here for compatibility, so that existing shell scripts invoking them can still function. For example, the compatibility package versions of `fastboot` and `fasthalt` do not prevent disk checks. It is recommended that the user consult the Manual Pages for the following commands, to find out about specific limitations: `fastboot`, `fasthalt`, `halt`, `reboot`, and `shutdown`.

`arch` Displays your machine architecture. This works the same as `uname -m` but `arch` is not in UNIX System V.

`fastboot` Reboots your machine. (See the `fastboot` compatibility package Manual Page.) Not in UNIX System V.

`fasthalt` Halts your machine. (See the `fasthalt` compatibility package Manual Page.) Not in UNIX System V.

`grpck` Checks that entries in the file `/etc/groups` are correct.

`halt` Shuts down the system. `halt` syncs the disks and writes out any information before shutting down. Not in UNIX System V, however, similar functionality is found in `shutdown` or `init 0`. (See the compatibility package Manual Page for `halt` for any limitations on BSD functionality.)

`hostid` Prints the numeric identifier (in hexadecimal) of the current host. Not in UNIX System V.

`hostname` Displays the name of your machine. Also allows you to rename it. Not in UNIX System V but it works like uname(1).

`install` Allows you to install (optional) software on your system. This version is very different from the UNIX System V one.

`logger` Allows you to add entries to a system log. Priorities, tags, and files to write to may be specified.

`mach` Reports the machine type.

`mach`'s functionality has been merged into `uname -p`; however, because many scripts and makefiles use `mach`, it has been placed in the compatibility package. Not in UNIX System V.

`mt` and `tcopy`

`mt` sends commands to a magnetic tape; `tcopy` copies a magnetic tape.

Both rely on a set of `ioctls` that are not present in default UNIX System V. However, users with new or enhanced device drivers may take advantage of this command. Not in UNIX System V.

`pagesize` Reports the size of a page in bytes. Useful for scripts. Not in UNIX System V.

`pwck` Similar to `grpck`, `pwck` checks to see that entries in your `passwd` file are correct.

`reboot` Reboots your system. Although not in UNIX System V, this command behaves like `shutdown` or `init 6`. (See the compatibility package Manual Page for `reboot` for any limitations on BSD functionality.)

`shutdown` This version of `shutdown` allows you to specify a time and a warning message to be sent. It also sends shutdown messages to other systems if they have mounted filesystems from the machine being shut down. (See the compatibility package Manual Page for `shutdown` for any limitations on BSD functionality.)

`uptime` Shows how long the system has been up. Not in UNIX System V.

```
$ uptime
 10:21pm  up 49 mins,  1 user
$
```

UNIX System V's `who -b` tells you when the system was last booted. BSD's `uptime` tells you how long it's been since the last boot. The two are equivalent.

Process Management

`lastcomm` Shows the last commands executed on the system. Also can be done by user or by terminal. Not in UNIX System V but `acctcom` is similar.

`ps` This version displays more information, in a different format, from that of UNIX System V. See the compatibility package `ps` Manual Page for details.

`renice` `renice`'s functionality is now present in the `priocntl` command; however, for BSD compatibility reasons, `renice` is included in the compatibility package. Not in UNIX System V.

User Information

`groups` Displays the groups to which a user belongs. Although the `id` command has been enhanced to perform this function, `groups` is included here for compatibility.

`users` A simple command that displays a short list of logged-in users. Not in UNIX System V. The UNIX System V equivalent is `who -q`.

`w` Much of `w`'s functionality exists in the 4.0 version of `whodo`; however, for full functionality, it is included here. Not in UNIX System V.

`whoami` Displays your effective user ID; `whoami` works even if you have used `su` to change your current user ID. Not in UNIX System V, although this is equivalent to `id`.

Miscellaneous

`printenv` Reports environment variables as currently set. Similar to the `env` command. Not in UNIX System V.

`stty` The Release 4.0 version of `stty` incorporates many of the features of BSD `stty`. The compatibility package version is included for full BSD functionality.

BSD Library Routines

Compatibility library routines are placed in one of the following four libraries: `libucb.a`, `libmp.a`, `libdbm.a`, and `libsocrpc.a`. (The ones in `librpcsoc.a` are covered in the *Transport Programming* manual. See also Chapter *overview*.

The routines found in these libraries are divided into two groups. The first group are those commands that are not included in UNIX System V Release 4.0 but which are useful enough to merit retaining for backward compatibility. The second group contains those routines that have counterparts under UNIX System V Release 4.0 but have differences in the user interfaces for those routines.

Routines Not Included in UNIX System V Release 4.0

The following routines do not exist in UNIX System V Release 4.0, but are included in a compatibility libraries for backwards compatibility purposes. At some later date these routines will no longer be supported and will eventually be discarded.

Routines in `libmp`

`libmp` contains the following routines: `gcd`, `itom`, `madd`, `mcmp`, `mdiv`, `mfree`, `min`, `mout`, `msqrt`, `msub`, `mtox`, `mult`, `pow`, `rpow`, `sdiv`. These routines perform arithmetic on integers of arbitrary length. They are contained in `/usr/ucblib/libmp.a`. For more details, please refer to the appropriate Manual Pages.

Routines in `libdbm`

`libdbm` contains the following routines: `dbmclose`, `dbminit`, `delete`, `fetch`, `firstkey`, `nextkey`, `store`. These routines maintain key/content pairs in a database using database routines. They are packaged in `/usr/ucb/libdbm.a`. For more information, see the appropriate Manual Pages.

Routines in `libucb`

`alloca`

Allocates the given number of bytes in the stack frame of the caller, and returns a pointer to the allocated block.

`re_comp` **and** `re_exec`

Similar to the UNIX System V routines `regcmp` and `regex`. `re_comp` compiles the regular expression in a string, returning 0 if it compiled successfully or returning a pointer to an error message if it did not. `regcmp`, on the other hand, returns a pointer to the compiled form or NULL if given an incorrect argument. `re_exec` returns 1 if the string *s* matches the last compiled regular expression, 0 if it fails to match, and -1 if the regular expression is invalid. `regex`, however, returns NULL on failure or a pointer to the next unmatched character on success.

See the LIBGEN library for UNIX System V equivalents.

`ualarm` **and** `usleep`

Simplified interfaces to the system calls handling high-resolution timers.

`initstate`, `random`, `setstate`, **and** `srandom`

Random number generator routines relied on by many user programs under BSD.

See `rand`(3C), `srand`(3C), and `drand`(3C) in the *Programmer's Reference Manual.*

`endusershell`, `getusershell`, **and** `setusershell`

Allow certain editing of the `passwd` file.

Database Manipulation Routines

The following routines maintain key/content pairs in a database using database routines that allow the maintaining of several databases at once: `dbm_clearerr`, `dbm_close`, `dbm_delete`, `dbm_error`, `dbm_fetch`, `dbm_firstkey`, `dbm_nextkey`, `dbm_open`, and `dbm_store`. For more details, please refer to the appropriate manual reference.

decimal_to_double, `decimal_to_extended`, and `decimal_to_single`

Convert decimal (integer) numbers to the appropriate floating point format. For more details, please refer to the appropriate manual reference. Corresponding conversion routines may be found in `decconv`(3C) in UNIX System V.

`single_to_decimal`, `double_to_decimal`, **and** `extended_to_decimal`

Convert floating point numbers in the appropriate formats to integers, rounding if necessary. For more details, please refer to the appropriate manual reference. Corresponding conversion routines may be found in `decconv`(3C) in UNIX System V.

`econvert`, `fconvert`, `gconvert`, `seconvert`, `sfconvert`, **and** `sgconvert`

Used to convert ASCII representations of numbers to various formats of float point numbers. For more details, please refer to the appropriate manual reference.

`ecvt`(3C), `fcvt`(3C), and `gcvt`(3C) are UNIX System V's closest equivalent's.

`ftime`

Returns a structure that contains elements showing the elapsed time since the epoch (00:00:00, January 1st, 1970). See `ctime`(3C).

`strcasecmp` **and** `strncasecmp`

Equivalent to `strcmp` and `strncmp` with the exception that the case of the characters is irrelevant.

`timezone`

Gets the time zone given an offset from GMT. `ctime`(3C) has many timezone features that will help with this functionality.

`alphasort` **and** `scandir`

These routines combine the functionalities of `readdir` and `qsort` to process directory information. `scandir` reads a directory and builds an array of pointers to directory entries, returning the number of entries in the array and a pointer to the array through the parameter, `namelist`. `namelist` is a pointer to an arrangement of directory structure pointers. If the directory cannot be opened for reading or if `malloc` cannot allocate enough memory to hold all the data structures, `scandir` returns `-1`. `alphasort` is a routine that alphabetically sorts an array of pointers to directory entries (built by the routine, `scandir`).

`bcmp`, `bcopy`, **and** `bzero`

Functionally equivalent to the ANSI functions, `memcmp`, `memcpy`, and `memset`. `bcopy` copies bytes, handling overlapping strings correctly. `bcmp` compares bytes, returning zero if they are identical, non-zero otherwise. `bzero` zeros out bytes.

`getwd`

Similar to `getcwd`. `getwd` returns the pathname of the current working directory, or NULL if there is an error. The return value of `getwd` is placed in a character string allocated by the caller.

`ieee_handler`

This provides easy exception handling to exploit ANSI/IEEE Std 754-1985 arithmetic in a C program. Results arising from invalid arguments and invalid combinations are undefined for efficiency.

`copysign`, `fp_class`, `isnan`, **and** `scalbn`

These functions provide capabilities required by ANSI/IEEE Standard 754-1985, or suggested in its appendix.

`copysign` copies the sign bit from one double to another; `fp_class` corresponds to IEEE's `class` and classifies doubles as zero, subnormal, NaN, and so on; `isnan` returns 1 if its argument is not a number; and `scalbn(x,n)` returns

```
x* 2**n
```

computed by exponent manipulation rather than by actually performing an exponentiation or a multiplication.

`fp_class` and `isnan` are equivalent to `isnan`(3C) and `scalban` works like `frexp`(3C).

`mkstemp`

Similar in function to `mktemp`(3C), but `mkstemp` returns the descriptor of the temporary file. `mkstemp` makes a unique filename and opens the file. Given a string that looks like a filename with six trailing `Xs`, `mkstemp` replaces the `Xs` with a letter and the current process ID. The letter is chosen so that the resulting name does not duplicate an existing file.

`index` **and** `rindex`

`index` returns a pointer to the first occurrence of a single character in a null-terminated character string. If the character does not occur in the string, `index` returns a NULL pointer. In UNIX System V, `string`(3C)'s `strchr` may be the appropriate replacement.

`rindex` returns a pointer to the last occurrence of a single character in a null-terminated string, or a NULL pointer if the letter does not occur in the string. In UNIX System V, `string`(3C)'s `strrchr` may be the appropriate replacement.

`setbuffer` and `setlinebuf`

These routines are used to provide control over the buffering used for input and output.

Similar to `setbuf` and `setvbuf`, `setbuffer` may be used after a stream has been opened but before it is read or written. It causes the array pointer to be used instead of an automatically allocated buffer. If the buffer is the NULL pointer, input/output is completely unbuffered.

`setlinebuf` is used to change the buffering on a stream from block buffered or unbuffered to line buffered. It can be used at any time that the file descriptor is active.

Routines Varying from UNIX System V Release 4.0

The following routines are located in `libucb.a` and differ slightly from UNIX System V:

`fopen`	UNIX System V and BSD versions differ in the handling of the `a` type. If mode `a` is specified, UNIX System V opens the file with `O_APPEND` set, while BSD doesn't. Also, BSD always seeks to the end of the file when mode `a` is specified, while UNIX System V seeks to the end of the file if update mode is not specified.
`nice`	BSD version always returns 0 if successful. The UNIX System V version returns the new `nice` value minus 20.
`nlist`	If unsuccessful, returns the number of symbols that were not located in the symbol table. (The UNIX System V version returns 0 if unsuccessful.) Also, `nlist` takes a filename as an argument, rather than a file descriptor.
`rand`	UNIX System V returns the generated number shifted to the right by 16 and anded with 0x7fff, while BSD returns the generated number anded with 0x7fffffff.

`sleep` UNIX System V returns the "unslept" amount, that is, the requested time minus the time actually slept. The BSD version does not return anything.

`times` BSD returns 0 when successful. UNIX System V returns the elapsed real time in clock ticks per second from an arbitrary point in the past.

`printf`, `fprintf`, `sprintf`, `vfprintf`, `vprintf`, and `vsprintf`

The following differences exist for these functions:

- In BSD, `sprintf` returns a pointer to its first argument; in UNIX System V, it returns the count of characters printed.
- In BSD, `printf` and `vprintf` always return the number of characters printed, while the UNIX System V version returns EOF in case of error.
- `fprintf` and `vfprintf` share the same differences as `printf` and `vprintf` (above); additionally, the buffering scheme for these routines is different for UNIX System V and BSD.

BSD System Calls and Signals

The following routines reside in `libucb.a`.

BSD System Calls

`getdtablesize`	Gets the size of a process's descriptor table. This function calls the UNIX System V system call `getrlimit`.
`gethostid`	Returns the 32-bit identifier for the current host. This function calls UNIX System V's `sysname`.
`gethostname`	Returns the standard hostname for the current processor. Calls the UNIX System V system call `sysname`.
`getrusage`	Gets information about system utilization. Calls UNIX System V's `times` system call.
`gettimeofday`	Get or set date and time. Calls the UNIX System V system call `hrtcntl`. Note that this version of `gettimeofday` accepts two arguments, although it doesn't use the second one.
`killpg`	Sends a signal to a process group. Calls `kill`, found in UNIX System V.
`reboot`	Restart system or halt processor. Calls the UNIX System V system call `uadmin`.
`setregid`	Set real and effective group ID's. Calls UNIX System V's `setuid` and `seteuid` system calls.
`setreuid`	Set real and effective user ID's. Calls UNIX System V's `setgid` and `setegid`.
`settimeofday`	Set date and time of day. Calls UNIX System V's system call `stime`. Note that this function accepts two arguments, although it ignores the second.
`syscall`	Performs indirect system calls. Calls the UNIX System V trap mechanism.
`utimes`	Sets file times. Calls the UNIX System V system call `utime`.

`wait3` Alternate system call (to `wait`) for waiting for a process to stop. This system call calls UNIX System V's `waitid` system call.

BSD Signals

BSD signal handlers and default actions are different from UNIX System V's.

The BSD style of signal handling is provided in the compatibility package. These routines invoke existing UNIX System V signal handling routines. These routines reside in `libucb.a`.

`_longjmp`, `longjmp`, `_setjmp`, and `setjmp`
In BSD, `setjmp` and `longjmp` save and restore the signal mask. This functionality has been preserved with the `sigsetjmp` and `siglongjmp` signals, respectively.

`_psignal` and `psignal`
In BSD, `_psignal` doesn't flush stdio buffers (neither does UNIX System V's version of `psignal`). The compatibility package version of `psignal`, however, does.

`sigblock` and `sigsetmask`
In BSD, `sigblock` blocks a signal; `sigsetmask` sets a signal mask. The functionality of these two has been folded into `sigprocmask` in the compatibility package .

`siginterrupt`, `signal`, `sigvec` and `sigaction`
In BSD, `siginterrupt` is a library function that allows signal interruptions; `signal` is a library function for simplified signal processing. Both now call `sigvec`, which itself calls the new compatibility signal handler `sigaction`.

`sigpause` Releases block signals; waits for interrupts. Implemented in the compatibility package through `sigsuspend`.

BSD Header Files

BSD Header Files Not Included in UNIX System V Release 4.0

Some BSD header files do not exist in UNIX System V Release 4.0. An attempt has been made to provide compatibility for header files, but in some cases it was impossible to achieve real source compatibility. The user should, wherever possible, be aware of the remaining differences.

`fp.h`	Used by the `econvert` and `decimal_to_string` family of routines. Note that this is equivalent to the `floatingpoint.h` header file. Generally, UNIX System V doesn't allow filenames of more than 14 characters; however, users who have installed the UFS file system may link `fp.h` to `floatingpoint.h` for compatibility.
`dbm.h`	Used by routines in `libdbm` and user programs invoking `libdbm`
`ndbm.h`	Used by `ndbm` routines and user programs invoking them.
`mp.h`	Used by `libmp` routines and user programs invoking them.
`strings.h`	Contains the same type of information as `string.h`.
`struct.h`	Contains useful macros.
`sunfp.h`	Used by the `decimal_to_string` family of routines. Note that this is the same as the `sunfloatingpoint.h` header file. Generally, UNIX System V doesn't allow filenames of more than 14 characters; however, users who have installed the UFS file system may link `sunfloatingpoint.h` to `sunfp.h` for compatibility.
`sysexits.h`	Contains useful exit codes.
`sys/ieeefp.h`	Used by the `econvert` and `decimal_to_string` family of routines, included by `floatingpoint.h`.
`sys/mtio.h`	Used by the `mt` and `tcopy` commands and the enhanced tape drivers.

`sys/reboot.h`	Used by the reboot system call wrapper in the Compatibility Package.
`ufs/quota.h`	Includes `sys/fs.ufs_quota.h`, and added for compatibility.

BSD Header Files Varying from UNIX System V Release 4.0

As with the library routines, there are some header files under UNIX System V Release 4.0 that differ slightly from their BSD counterparts. In those instances, the affected header files are included in this compatibility package with as many additions as possible to keep them compatible with BSD. The following list shows those files affected and the reasons for their inclusion in this package.

`assert.h`	Included for compatibility.
`regexp.h`	Includes `ctype.h` for compatibility.
`unistd.h`	Includes `#defines` missing in the UNIX System V Release 4.0 version of this file.
`setjmp.h`	Includes additional `#defines` and `#includes` for compatibility.
`signal.h`	Contains `sys/signal.h`. Included here for compatibility.
`stdio.h`	Includes BSD-specific definitions.
`sys/dirent.h`	Includes the missing DIRSIZ macro used by many BSD utilities.
`sys/fcntl.h`	Includes missing `#defines`.
`sys/file.h`	Includes missing `#defines` such as `L_SET`, `L_INCR`.
`sys/param.h`	Includes missing `#defines`.
`sys/resource.h`	Includes the `rusage` structure missing from the UNIX System V Release 4.0 version of `resource.h` and used by the `getrusage` system call wrapper.

`sys/signal.h`	Includes BSD-specific data structures and definitions.
`sys/types.h`	Includes the `sys/sysmacros` header files and additional type definitions.
`sys/vfs.h`	Includes `sys/statfs` header file.
`sys/wait.h`	Includes declarations required by the `wait3` system call wrapper.

Table of Contents

1. Commands

3. Functions

4. File Formats

7. Special Files

Permuted Index

NAME

addbib – create or extend a bibliographic database

SYNOPSIS

/usr/ucb/addbib [-a] [-p *promptfile*] *database*

DESCRIPTION

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

The following options are available:

-a Suppress prompting for an abstract; asking for an abstract is the default. Abstracts are ended with a CTRL–D.

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

USAGE

Bibliography Key Letters

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

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

%N	Number within volume
%O	Other commentary, printed at end of reference
%P	Page number(s)
%Q	Corporate or Foreign Author (unreversed)
%R	Report, paper, or thesis (unpublished)
%S	Series title
%T	Title of article or book
%V	Volume number
%X	Abstract — used by roffbib, not by refer
%Y, Z	Ignored by refer

SEE ALSO

indxbib(1), lookbib(1), refer(1), roffbib(1), sortbib(1), ed(1), ex(1), vi(1) in the *User's Reference Manual.*

NAME

apropos – locate commands by keyword lookup

SYNOPSIS

`/usr/ucb/apropos` *keyword...*

DESCRIPTION

`apropos` shows which manual sections contain instances of any of the given keywords in their title. Each word is considered separately and the case of letters is ignored. Words which are part of other words are considered; thus, when looking for 'compile', `apropos` will find all instances of 'compiler' also.

Try

```
apropos password
```

and

```
apropos editor
```

If the line starts '*filename*(*section*) ...' you can do '`man` *section filename*' to get the documentation for it. Try

```
apropos format
```

and then

```
man 3s printf
```

to get the manual page on the subroutine `printf`.

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

FILES

`/usr/share/man/whatis` data base

SEE ALSO

`man`(1), `whatis`(1), `catman`(1M).

NAME

arch – display the architecture of the current host

SYNOPSIS

/usr/ucb/arch

DESCRIPTION

The arch command displays the architecture of the current host system.

SEE ALSO

mach(1).

uname(1) in the *User's Reference Manual.*

NAME

basename – display portions of pathnames

SYNOPSIS

/usr/ucb/basename *string* [*suffix*]

DESCRIPTION

basename deletes any prefix ending in '/' and the *suffix*, if present in *string*. It directs the result to the standard output, and is normally used inside substitution marks (` `) within shell procedures. The *suffix* is a pattern as defined on the ed(1) manual page.

EXAMPLE

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

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

SEE ALSO

ed(1), sh(1) in the *User's Reference Manual.*

NAME

biff – give notice of incoming mail messages

SYNOPSIS

/usr/ucb/biff [y | n]

DESCRIPTION

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

The y option allows mail notification for the terminal. The n option disables notification for the terminal.

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

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

FILES

~/.login
~/.profile

SEE ALSO

csh(1), mail(1), sh(1) in the *User's Reference Manual.*

NAME

cc – C compiler

SYNOPSIS

`/usr/ucb/cc` [*options*]

DESCRIPTION

`/usr/ucb/cc` is the C compiler for the BSD Compatibility Package. `/usr/ucb/cc` is identical to `/usr/bin/cc` (see cc(1)) except that BSD header files are used BSD libraries are linked *before* System V libraries.

`/usr/ucb/cc` accepts the same options as `/usr/bin/cc`, with the following exceptions:

`-I`*"dir"*

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

`-L`*"dir"*

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

`-Y LU,`*dir*

Change the default directory used for finding libraries.

FILES

`/usr/ucblib`
`/usr/lib/ld`
`/usr/ucblib/libucb.a`
`/usr/lib/libucb.a`

NOTES

The `-Y LU,`*dir* option may have unexpected results, and should not be used. This option is not in the UNIX System V base.

SEE ALSO

`ld(1)`.

`as(1)`, `ar(1)`, `cc(1)`, `ld(1)`, `lorder(1)`, `ranlib(1)`, `strip(1)`, `tsort(1)`, `a.out(4)` in the *Programmer's Reference Manual.*

NAME

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

SYNOPSIS

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

DESCRIPTION

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

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

checknr knows about the ms and me macro packages.

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

The following options are available:

-f Ignore \f font changes.

-s Ignore \s size changes.

-a *.x1 .y1...*
Add pairs of macros to the list. The pairs of macros are assumed to be those (such as .DS and .DE) that should be checked for balance. The -a option must be followed by groups of six characters, each group defining a pair of macros. The six characters are a period, the first macro name, another period, and the second macro name. For example, to define a pair .BS and .ES, use -a.BS.ES

-c *.x1...*
Define commands which checknr would otherwise complain about as undefined.

SEE ALSO

eqn(1), nroff(1), troff(1), me(7), ms(7).

NOTES

There is no way to define a one-character macro name using the -a option.

NAME

chown – change file owner

SYNOPSIS

`/usr/ucb/chown` [`-fhR`] *owner*[*.group*] *file ...*

DESCRIPTION

`chown` changes the owner of the *files* to *owner*. The *owner* may be either a decimal user ID or a login name found in `/etc/passwd` file. The optional *.group* suffix may be used to change the group at the same time.

If `chown` is invoked by other than the super-user, the set-user-ID bit of the file mode, 04000, is cleared.

Only the super-user may change the owner of a file.

Valid options to `chown` are:

`-f` Suppress error reporting

`-h` If the file is a symbolic link, change the owner of the symbolic link. Without this option, the owner of the file referenced by the symbolic link is changed.

`-R` Descend recursively through directories setting the ownership ID of all files in each directory entered.

FILES

`/etc/group`
`/etc/passwd`

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 the Remote File Sharing chapter of the *System Administrator's Guide.*

SEE ALSO

`chgrp`(1), `chmod`(1)

`chgrp`(1), `chmod`(1) in the *User's Reference Manual.*
`chown`(2) in the *Programmer's Reference Manual.*
`passwd`(4) in the *System Administrator's Reference Manual.*

NAME

deroff – remove nroff, troff, tbl and eqn constructs

SYNOPSIS

/usr/ucb/deroff [-w] *filename* . . .

DESCRIPTION

The deroff command reads each file in sequence and removes all nroff and troff command lines, backslash constructions, macro definitions, eqn constructs (between .EQ and .EN lines or between delimiters), and table descriptions and writes the remainder on the standard output. deroff follows chains of included files (.so and .nx commands); if a file has already been included, a .so is ignored and a .nx terminates execution. If no input file is given, deroff reads from the standard input file.

OPTIONS

-w Generate a word list, one word per line. A 'word' is a string of letters, digits, and apostrophes, beginning with a letter; apostrophes are removed. All other characters are ignored.

SEE ALSO

eqn(1), nroff(1), tbl(1), troff(1)

NOTES

deroff is not a complete troff interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

deroff does not work well with files that use .so to source in the standard macro package files.

NAME

df – report free disk space on file systems

SYNOPSIS

/usr/ucb/df [-beglntvV] [*directory* | *special*]

DESCRIPTION

df displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used.

If arguments to df are path names, df produces a report on the file system containing the named file. Thus 'df .' shows the amount of space on the file system containing the current directory.

The following options are available:

-b Print only the number of kilobytes free.

-e Print only the number of file entries free.

-g Print the entire statvfs structure. Not valid with any other option, except the -l option.

-l Report on local file systems only.

-n Print only the file system type name.

-t Report figures for total allocated blocks and files as well as free blocks and files.

-v Print allocations in kilobytes.

-V Verbose. Echo complete command line.

EXAMPLE

A sample of output for df looks like:

```
df
Filesystem  kbytes  used  avail  capacity  Mounted on
sparky:/      7445   4714 1986   70%       /
sparky:/usr 42277   35291 2758   93%       /usr
```

FILES

/etc/mnttab	list of file systems currently mounted
/etc/vfstab	list of default parameters for each file system

SEE ALSO

du(1M), quot(1M), tunefs(1M), mnttab(4) in the *System Administrator's Reference Manual.*

NAME

diffmk – mark differences between versions of a **troff** input file

SYNOPSIS

/usr/ucb/diffmk *oldfile newfile markedfile*

DESCRIPTION

The **diffmk** command compares two versions of a file and creates a third version that includes "change mark" (**.mc**) commands for **nroff** and **troff**. *oldfile* and *newfile* are the old and new versions of the file. **diffmk** generates *markedfile*, which, contains the text from *newfile* with **troff**(1) "change mark" requests (**.mc**) inserted where *newfile* differs from *oldfile*. When *markedfile* is formatted, changed or inserted text is shown by a **|** at the right margin of each line. The position of deleted text is shown by a single *.

diffmk can also be used in conjunction with the proper **troff** requests to produce program listings with marked changes. In the following command line:

```
diffmk old.c new.c marked.c ; nroff reqs marked.c | pr
```

the file **reqs** contains the following **troff** requests:

```
.pl 1
.ll 77
.nf
.eo
.nh
```

which eliminate page breaks, adjust the line length, set no-fill mode, ignore escape characters, and turn off hyphenation, respectively.

If the characters **|** and ***** are inappropriate, you might run *markedfile* through sed to globally change them.

SEE ALSO

nroff(1), **troff**(1)

diff(1), sed(1) in the *User's Reference Manual.*

NOTES

Aesthetic considerations may dictate manual adjustment of some output. File differences involving only formatting requests may produce undesirable output, that is, replacing **.sp** by **.sp 2** will produce a "change mark" on the preceding or following line of output.

NAME

echo – echo arguments

SYNOPSIS

/usr/ucb/echo [arg] ...
/usr/ucb/echo [-n] [arg]

DESCRIPTION

echo writes its arguments separated by blanks and terminated by a new-line on the standard output.

The /usr/bin/sh version understands the following 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
\0*n*	where *n* is the 8-bit character whose ASCII code is the 1-, 2- or 3-digit octal number representing that character.

The following option is available to /usr/bin/sh users only if /usr/ucb preceds /usr/bin in the user's PATH. It is available to /usr/csh users, regardless of PATH:

-n	Do not add the newline to the output.

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

SEE ALSO

sh(1) in the *User's Reference Manual.*

NOTES

The -n option is a transition aid for BSD applications, and may not be supported in future releases.

The When representing an 8-bit character by using the escape convention \0*n*, 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

eqn, neqn, checkeq – typeset mathematics

SYNOPSIS

/usr/ucb/eqn [**-d***xy*] [**-f***n*] [**-p***n*] [**-s***n*] [*filename*] ...

/usr/ucb/neqn [*filename*] ...

/usr/ucb/checkeq [*filename*] ...

DESCRIPTION

The **eqn** and **neqn** commands are language processors to assist in describing equations. **eqn** is a preprocessor for **troff**(1) and is intended for devices that can print **troff**'s output. **neqn** is a preprocessor for **nroff**(1) and is intended for use with terminals.

checkeq reports missing or unbalanced delimiters and **.EQ**/**.EN** pairs.

If no *filename*s are specified, **eqn** and **neqn** read from the standard input. A line beginning with **.EQ** marks the start of an equation; the end of an equation is marked by a line beginning with **.EN**. Neither of these lines is altered, so they may be defined in macro packages to get centering, numbering, etc. It is also possible to set two characters as "delimiters"; subsequent text between delimiters is also treated as **eqn** input.

The following options are available for **eqn** and **neqn**:

-d*xy* Set equation delimiters set to characters *x* and *y* with the command-line argument. The more common way to do this is with **delim***xy* between **.EQ** and **.EN**. The left and right delimiters may be identical. Delimiters are turned off by **delim off** appearing in the text. All text that is neither between delimiters nor between **.EQ** and **.EN** is passed through untouched.

-f*n* Change font to *n* globally in the document. The font can also be changed globally in the body of the document by using the **gfont** directive.

-p*n* Reduce subscripts and superscripts by *n* point sizes from the previous size. In the absence of the **-p** option, subscripts and superscripts are reduced by 3 point sizes from the previous size.

-s*n* Set equations in point size *n* globally in the document. The point size can also be changed globally in the body of the document by using the **gsize** directive.

-T*dev* Prepare output for device *dev*. If no **-T** option is present, **eqn** looks at the environment variable **TYPESETTER** to see what the intended output device is. If no such variable is found in the environment, a system-dependent default device is assumed. Not available using **neqn**.

USAGE

eqn Language

Tokens within **eqn** are separated by braces, double quotes, tildes, circumflexes, SPACE, TAB, or NEWLINE characters. Braces { } are used for grouping; generally speaking, anywhere a single character like *x* could appear, a complicated construction enclosed in braces may be used instead. Tilde (~) represents a full SPACE in the output, circumflex (^) half as much.

Subscripts and superscripts are produced with the keywords `sub` and `sup`. Thus '`x sub i`' makes x_i, '`a sub i sup 2`' produces a_i^2, and '`e sup {x sup 2 + y sup 2}`' gives $e^{x^2+y^2}$.

Fractions are made with `over`: '`a over b`' yields $\frac{a}{b}$.

`sqrt` makes square roots: '`1 over down 10 sqrt {ax sup 2 +bx+c}`' results in

$$\frac{1}{\sqrt{ax^2+bx+c}}.$$

Although `eqn` tries to get most things at the right place on the paper, occasionally you will need to tune the output to make it just right. In the previous example, a local motion, *down 10* was used to get more space between the square root and the line above it.

The keywords `from` and `to` introduce lower and upper limits on arbitrary things: $\lim_{n\to\infty} \sum_{0}^{n} x_i$ is made with '`lim from {n-> inf } sum from 0 to n x sub i`'.

Left and right brackets, braces, etc., of the right height are made with `left` and `right`: '`left [ x sup 2 + y sup 2 over alpha right ] ~=~1`' produces

$$\left[x^2+\frac{y^2}{\alpha}\right] = 1.$$

The `right` clause is optional. Legal characters after `left` and `right` are braces, brackets, bars, `c` and `f` for ceiling and floor, and "" for nothing at all (useful for a right-side-only bracket).

Vertical piles of things are made with `pile`, `lpile`, `cpile`, and `rpile`: '`pile {a above b above c}`' produces $\begin{matrix} a \\ b \\ c \end{matrix}$. There can be an arbitrary number of elements in a pile. `lpile` left-justifies, `pile` and `cpile` center, with different vertical spacing, and `rpile` right justifies.

Matrices are made with `matrix`: '`matrix { lcol { x sub i above y sub 2 } ccol { 1 above 2 } }`' produces $\begin{matrix} x_i & 1 \\ y_2 & 2 \end{matrix}$. In addition, there is `rcol` for a right-justified column.

Diacritical marks are made with `dot`, `dotdot`, `hat`, `tilde`, `bar`, `vec`, `dyad`, and `under`: '`x dot = f(t) bar`' is $\dot{x}=\overline{f(t)}$, '`y dotdot bar ~=~ n under`' is $\ddot{\bar{y}} = \underline{n}$, and '`x vec ~=~ y dyad`' is $\vec{x} = \overleftrightarrow{y}$.

Sizes and font can be changed with `size` *n* or `size` ±*n*, `roman`, `italic`, `bold`, and `font` *n*. Size and fonts can be changed globally in a document by `gsize` *n* and `gfont` *n*, or by the command-line arguments `-s`*n* and `-f`*n*.

Successive display arguments can be lined up. Place `mark` before the desired lineup point in the first equation; place `lineup` at the place that is to line up vertically in subsequent equations.

Shorthands may be defined or existing keywords redefined with `define`:

`define` *thing* `%` *replacement* `%`

defines a new token called *thing* which will be replaced by *replacement* whenever it appears thereafter. The `%` may be any character that does not occur in *replacement*.

Keywords like `sum` (∑), `int` (∫), `inf` (∞), and shorthands like `>=` (≥), `->` (→), and `!=` (≠) are recognized. Greek letters are spelled out in the desired case, as in `alpha` or `GAMMA`. Mathematical words like sin, cos, and log are made Roman automatically. `troff`(1) four-character escapes like `\(bu` (•) can be used anywhere. Strings enclosed in double quotes "..." are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with `troff` when all else fails.

EXAMPLE

```
eqn filename . . . | troff
neqn filename ... | nroff
```

SEE ALSO

`tbl`(1), `troff`(1), eqnchar(7), `ms`(7)

NOTES

To embolden digits, parens, etc., it is necessary to quote them, as in `bold "12.3"`.

NAME

fsirand – install random inode generation numbers

SYNOPSIS

/usr/ucb/fsirand [-p] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special*, and also installs a filesystem ID in the superblock. This helps increase the security of filesystems exported by NFS.

fsirand must be used only on an unmounted filesystem that has been checked with fsck(1M). The only exception is that it can be used on the root filesystem in single-user mode, if the system is immediately re-booted afterwords.

The -p option prints out the generation numbers for all the inodes, but does not change the generation numbers.

SEE ALSO

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

NAME

groups – display a user's group memberships

SYNOPSIS

/usr/ucb/groups [*user* ...]

DESCRIPTION

With no arguments, groups displays the groups to which you belong; else it displays the groups to which the user belongs. Each user belongs to a group specified in the password file /etc/passwd and possibly to other groups as specified in the file /etc/group. If you do not own a file but belong to the group which it is owned by then you are granted group access to the file.

FILES

/etc/passwd
/etc/group

SEE ALSO

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

NOTES

This command is obsolescent.

NAME

hostid – print the numeric identifier of the current host

SYNOPSIS

/usr/ucb/hostid

DESCRIPTION

The hostid command prints the identifier of the current host in hexadecimal. This numeric value is likely to differ when hostid is run on a different machine.

SEE ALSO

gethostid(2)
sysinfo(2) in the *Programmer's Reference Manual.*

NAME

hostname – set or print name of current host system

SYNOPSIS

/usr/ucb/hostname [*name-of-host*]

DESCRIPTION

The hostname command prints the name of the current host, as given before the login prompt. The super-user can set the hostname by giving an argument.

SEE ALSO

uname(1) in the *User's Reference Manual.*

NAME

indxbib – create an inverted index to a bibliographic database

SYNOPSIS

/usr/ucb/indxbib *database-file...*

DESCRIPTION

indxbib makes an inverted index to the named *database-file* (which must reside within the current directory), typically for use by lookbib and refer. A *database* contains bibliographic references (or other kinds of information) separated by blank lines.

A bibliographic reference is a set of lines, constituting fields of bibliographic information. Each field starts on a line beginning with a '%', followed by a key-letter, then a blank, and finally the contents of the field, which may continue until the next line starting with '%' (see addbib).

indxbib is a shell script that calls two programs: mkey and inv. mkey truncates words to 6 characters, and maps upper case to lower case. It also discards words shorter than 3 characters, words among the 100 most common English words, and numbers (dates) < 1900 or > 2000. These parameters can be changed.

indxbib creates an entry file (with a .ia suffix), a posting file (.ib), and a tag file (.ic), in the working directory.

FILES

/usr/ucblib/reftools/mkey
/usr/ucblib/reftools/inv

*.ia	entry file
*.ib	posting file
*.ic	tag file
*.ig	reference file

SEE ALSO

addbib(1), lookbib(1), refer(1), roffbib(1), sortbib(1).

NOTES

All dates should probably be indexed, since many disciplines refer to literature written in the 1800s or earlier.

indxbib does not recognize pathnames.

NAME

install – install files

SYNOPSIS

/usr/ucb/install [-cs] [-g *group*] [-m *mode*] [-o *owner*] *file1 file2*

/usr/ucb/install [-cs] [-g *group*] [-m *mode*] [-o *owner*] *file...directory*

/usr/ucb/install -d [-g *group*] [-m *mode*] [-o *owner*] *directory*

DESCRIPTION

Install is used within makefiles to copy new versions of files into a destination directory and to create the destination directory itself.

The first two forms are similar to the cp(1) command with the addition that executable files can be stripped during the copy and the owner, group, and mode of the installed file(s) can be given.

The third form can be used to create a destination directory with the required owner, group and permissions.

Note: install uses no special privileges to copy files from one place to another. The implications of this are:

- You must have permission to read the files to be installed.
- You must have permission to copy into the destination file or directory.
- You must have permission to change the modes on the final copy of the file if you want to use the -m option to change modes.
- You must be superuser if you want to specify the ownership of the installed file with -o. If you are not the super-user, or if -o is not in effect, the installed file will be owned by you, regardless of who owns the original.

OPTIONS

-g *group* Set the group ownership of the installed file or directory. (staff by default)

-m *mode* Set the mode for the installed file or directory. (0755 by default)

-o *owner* If run as root, set the ownership of the installed file to the user-ID of *owner*.

-c Copy files. In fact install *always* copies files, but the -c option is retained for backwards compatibility with old shell scripts that might otherwise break.

-s Strip executable files as they are copied.

-d Create a directory. Missing parent directories are created as required as in mkdir -p. If the directory already exists, the owner, group and mode will be set to the values given on the command line.

SEE ALSO

chown(1)

chgrp(1), chmod(1), cp(1), mkdir(1), strip(1) in the *User's Reference Manual.*
install(1M) in the *System Administrator's Reference Manual.*

NAME

`lastcomm` – show the last commands executed, in reverse order

SYNOPSIS

`/usr/ucb/lastcomm` [*command-name*] ... [*user-name*] ... [*terminal-name*] ...

DESCRIPTION

The `lastcomm` command gives information on previously executed commands. `lastcomm` with no arguments displays information about all the commands recorded during the current accounting file's lifetime. If called with arguments, `lastcomm` only displays accounting entries with a matching *command-name*, *user-name*, or *terminal-name*.

EXAMPLE

The command:

```
lastcomm a.out root term/01
```

would produce a listing of all the executions of commands named `a.out`, by user `root` while using the terminal `term/01`. and

```
lastcomm root
```

would produce a listing of all the commands executed by user `root`.

For each process entry, `lastcomm` displays the following items of information:

- The command name under which the process was called.
- One or more flags indicating special information about the process. The flags have the following meanings:
  - `F` The process performed a `fork` but not an `exec`.
  - `S` The process ran as a set-user-id program.
- The name of the user who ran the process.
- The terminal which the user was logged in on at the time (if applicable).
- The amount of CPU time used by the process (in seconds).
- The date and time the process exited.

FILES

`/var/adm/pacct` accounting file

SEE ALSO

`sigvec`(3)

`last`(1) in the *User's Reference Manual.*
`acct`(4), `core`(4) in the *System Administrator's Reference Manual.*

NAME

ld – link editor, dynamic link editor

SYNOPSIS

/usr/ucb/ld [*options*]

DESCRIPTION

/usr/ucb/ld is the link editor for the BSD Compatibility Package. /usr/ucb/ld is identical to /usr/bin/ld [see ld(1)] except that BSD libraries and routines are included *before* System V libraries and routines.

/usr/ucb/ld accepts the same options as /usr/bin/ld, with the following exceptions:

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

-Y LU,*dir*
Change the default directory used for finding libraries. Warning: this option may have unexpected results, and should not be used.

FILES

/usr/ucblib
/usr/lib
/usr/ucblib/libx.a
/usr/lib/libx.a

SEE ALSO

ar(1), as(1), cc(1), ld(1), lorder(1), strip(1), tsort(1) in the *Programmer's Reference Manual.*

NAME

ln – make hard or symbolic links to files

SYNOPSIS

/usr/ucb/ln [-fs] *filename* [*linkname*]
/usr/ucb/ln [-fs] *pathname*... *directory*

DESCRIPTION

/usr/ucb/ln creates an additional directory entry, called a link, to a file or directory. Any number of links can be assigned to a file. The number of links does not affect other file attributes such as size, protections, data, etc.

filename is the name of the original file or directory. *linkname* is the new name to associate with the file or filename. If *linkname* is omitted, the last component of *filename* is used as the name of the link.

If the last argument is the name of a directory, symbolic links are made in that directory for each *pathname* argument; /usr/ucb/ln uses the last component of each *pathname* as the name of each link in the named *directory*.

A hard link (the default) is a standard directory entry just like the one made when the file was created. Hard links can only be made to existing files. Hard links cannot be made across file systems (disk partitions, mounted file systems). To remove a file, all hard links to it must be removed, including the name by which it was first created; removing the last hard link releases the inode associated with the file.

A symbolic link, made with the -s option, is a special directory entry that points to another named file. Symbolic links can span file systems and point to directories. In fact, you can create a symbolic link that points to a file that is currently absent from the file system; removing the file that it points to does not affect or alter the symbolic link itself.

A symbolic link to a directory behaves differently than you might expect in certain cases. While an ls(1V) on such a link displays the files in the pointed-to directory, an 'ls -l' displays information about the link itself:

```
example% /usr/ucb/ln -s dir link
example% ls link
file1 file2 file3 file4
example% ls -l link
lrwxrwxrwx  1 user            7 Jan 11 23:27 link -> dir
```

When you cd(1) to a directory through a symbolic link, you wind up in the pointed-to location within the file system. This means that the parent of the new working directory is not the parent of the symbolic link, but rather, the parent of the pointed-to directory. For instance, in the following case the final working directory is /usr and not /home/user/linktest.

```
example% pwd
/home/user/linktest
example% /usr/ucb/ln -s /var/tmp symlink
example% cd symlink
example% cd ..
example% pwd
/usr
```

C shell user's can avoid any resulting navigation problems by using the pushd and popd built-in commands instead of cd.

OPTIONS

-f Force a hard link to a directory — this option is only available to the super-user.

-s Create a symbolic link or links.

EXAMPLE

The commands below illustrate the effects of the different forms of the /usr/ucb/ln command:

```
example% /usr/ucb/ln file link
example% ls -F file link
file   link
example% /usr/ucb/ln -s file symlink
example% ls -F file symlink
file   symlink@
example% ls -li file link symlink
 10606 -rw-r--r--  2 user            0 Jan 12 00:06 file
 10606 -rw-r--r--  2 user            0 Jan 12 00:06 link
 10607 lrwxrwxrwx  1 user            4 Jan 12 00:06 symlink -> file
example% /usr/ucb/ln -s nonesuch devoid
example% ls -F devoid
devoid@
example% cat devoid
devoid: No such file or directory
example% /usr/ucb/ln -s /proto/bin/* /tmp/bin
example% ls -F /proto/bin /tmp/bin
/proto/bin:
x*      y*      z*

/tmp/bin:
x@      y@      z@
```

SEE ALSO

cp(1), ls(1), mv(1), rm(1) in the *User's Reference Manual.*

link(2), readlink(2), stat(2), symlink(2) in the *Programmer's Reference Manual.*

NOTES

When the last argument is a directory, simple basenames should not be used for *pathname* arguments. If a basename is used, the resulting symbolic link points to itself:

```
example% /usr/ucb/ln -s file /tmp
example% ls -l /tmp/file
lrwxrwxrwx  1 user            4 Jan 12 00:16 /tmp/file -> file
example% cat /tmp/file
/tmp/file: Too many levels of symbolic links
```

To avoid this problem, use full pathnames, or prepend a reference to the PWD variable to files in the working directory:

```
example% rm /tmp/file
example% /usr/ucb/ln -s $PWD/file /tmp
lrwxrwxrwx  1 user
      4 Jan 12 00:16 /tmp/file -> /home/user/subdir/file
```

NAME

logger – add entries to the system log

SYNOPSIS

/usr/ucb/logger [-t *tag*] [-p *priority*] [-i] [-f *filename*] [*message*] ...

DESCRIPTION

logger provides a method for adding one-line entries to the system log file from the command line. One or more *message* arguments can be given on the command line, in which case each is logged immediately. Otherwise, a *filename* can be specified, in which case each line in the file is logged. If neither is specified, logger reads and logs messages on a line-by-line basis from the standard input.

The following options are available:

-t *tag*	Mark each line added to the log with the specified *tag*.
-p *priority*	Enter the message with the specified *priority*. The message priority can be specified numerically, or as a *facility.level* pair. For example, '-p local3.info' assigns the message priority to the info level in the local3 facility. The default priority is user.notice.
-i	Log the process ID of the logger process with each line.
-f *filename*	Use the contents of *filename* as the message to log.
message	If this is unspecified, either the file indicated with -f or the standard input is added to the log.

EXAMPLE

```
logger System rebooted
```

will log the message 'System rebooted' to the facility at priority notice to be treated by syslogd as other messages to the facility notice are.

```
logger -p local0.notice -t HOSTIDM -f /dev/idmc
```

will read from the file /dev/idmc and will log each line in that file as a message with the tag 'HOSTIDM' at priority notice to be treated by syslogd as other messages to the facility local0 are.

SEE ALSO

syslog(3), syslogd(1M)

NAME

look – find words in the system dictionary or lines in a sorted list

SYNOPSIS

/usr/ucb/look [-d] [-f] [-tc] *string* [*filename*]

DESCRIPTION

The look command consults a sorted *filename* and prints all lines that begin with *string*.

If no *filename* is specified, look uses /usr/ucblib/dict/words with collating sequence -df.

The following options are available:

-d Dictionary order. Only letters, digits, TAB and SPACE characters are used in comparisons.

-f Fold case. Upper case letters are not distinguished from lower case in comparisons.

-tc Set termination character. All characters to the right of *c* in *string* are ignored.

FILES

/usr/ucblib/dict/words

SEE ALSO

grep(1), sort(1) in the *User's Reference Manual.*

NAME

lookbib – find references in a bibliographic database

SYNOPSIS

/usr/ucb/lookbib *database*

DESCRIPTION

A bibliographic reference is a set of lines, constituting fields of bibliographic information. Each field starts on a line beginning with a '%', followed by a key-letter, then a blank, and finally the contents of the field, which may continue until the next line starting with '%'. See addbib.

lookbib uses an inverted index made by indxbib to find sets of bibliographic references. It reads keywords typed after the '>' prompt on the terminal, and retrieves records containing all these keywords. If nothing matches, nothing is returned except another '>' prompt.

It is possible to search multiple databases, as long as they have a common index made by indxbib. In that case, only the first argument given to indxbib is specified to lookbib.

If lookbib does not find the index files (the .i[abc] files), it looks for a reference file with the same name as the argument, without the suffixes. It creates a file with a .ig suffix, suitable for use with fgrep (see grep). lookbib then uses this fgrep file to find references. This method is simpler to use, but the .ig file is slower to use than the .i[abc] files, and does not allow the use of multiple reference files.

FILES

*.ia *.ib *.ic	index files
*.ig	reference file

SEE ALSO

addbib(1), indxbib(1), refer(1), roffbib(1), sortbib(1).

grep(1) in the *User's Reference Manual.*

NOTES

Probably all dates should be indexed, since many disciplines refer to literature written in the 1800s or earlier.

NAME

lpq – display the queue of printer jobs

SYNOPSIS

/usr/ucb/lpq [-P*printer*] [-l] [+ [*interval*]] [*job#* ...] [*username* ...]

DESCRIPTION

lpq displays the contents of a printer queue. It reports the status of jobs specified by *job#*, or all jobs owned by the user specified by *username*. lpq reports on all jobs in the default printer queue when invoked with no arguments.

For each print job in the queue, lpq reports the user's name, current position, the names of input files comprising the job, the job number (by which it is referred to when using lprm(1)) and the total size in bytes. Normally, only as much information as will fit on one line is displayed. Jobs are normally queued on a first-in-first-out basis. Filenames comprising a job may be unavailable, such as when lpr is used at the end of a pipeline; in such cases the filename field indicates the standard input.

If lpq warns that there is no daemon present (that is, due to some malfunction), the lpc(1M) command can be used to restart a printer daemon.

OPTIONS

-P *printer* — Display information about the queue for the specified *printer*. In the absence of the -P option, the queue to the printer specified by the PRINTER variable in the environment is used. If the PRINTER variable is not set, the queue for the default printer is used.

-l — Display queue information in long format; includes the name of the host from which the job originated.

+[*interval*] — Display the spool queue periodically until it empties. This option clears the terminal screen before reporting on the queue. If an *interval* is supplied, lpq sleeps that number of seconds in between reports.

FILES

/var/spool/lp — spooling directory.

/var/spool/lp/tmp/*system_name*/*-0 — reqeust files specifying jobs

DIAGNOSTICS

printer is printing

The lpq program queries the spooler LPSCHED about the status of the printer. If the printer is disabled, the superuser can restart the spooler using lpc(1M).

printer waiting for auto-retry (offline ?)

The daemon could not open the printer device. The printer may be turned off-line. This message can also occur if a printer is out of paper, the paper is jammed, and so on. Another possible cause is that a process, such as an output filter, has exclusive use of the device. The only recourse in this case is to kill the offending process and restart the printer with lpc.

`waiting for` *host* `to come up`
A daemon is trying to connect to the remote machine named *host*, in order to send the files in the local queue. If the remote machine is up, `lpd` on the remote machine is probably dead or hung and should be restarted using `lpc`.

`sending to` *host*
The files are being transferred to the remote *host*, or else the local daemon has hung while trying to transfer the files.

`printer disabled reason:`
The printer has been marked as being unavailable with `lpc`.

`lpq: The LP print service isn't running or can't be reached.`
The `lpsched` process overseeing the spooling queue does not exist. This normally occurs only when the daemon has unexpectedly died. You can restart the printer daemon with `lpc`.

`lpr:` *printer*`: unknown printer`
The `printer` was not found in the System V LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use '`lptstat -p`' to find the reason.

`lpr: error on opening queue to spooler`
The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon `/usr/lib/lpsched` is running.

`lpr: Can't send message to LP print service`

`lpr: Can't receive message from LP print service`
These indicate that the LP print service has been stopped. Get help from the system administrator.

`lpr: Received unexpected message from LP print service`
It is likely there is an error in this software. Get help from system administrator.

SEE ALSO

`lpc`(1M), `lpr`(1), `lprm`(1),

`lpsched`(1M) in the *System Administrator's Reference Manual.*
`lp`(1) in the *User's Reference Manual.*

NOTES

Output formatting is sensitive to the line length of the terminal; this can result in widely-spaced columns.

NAME

lpr – send a job to the printer

SYNOPSIS

/usr/ucb/lpr [-P *printer*] [-# *copies*] [-C *class*] [-J *job*] [-T *title*]
[-i [*indent*]] [-w *cols*] [-B] [-r] [-m] [-h] [-s]
[*-filter_option*] [*filename* ...]

DESCRIPTION

lpr forwards printer jobs to a spooling area for subsequent printing as facilities become available. Each printer job consists of copies of, or, with -s , complete pathnames of each *filename* you specify. The spool area is managed by the line printer spooler, lpsched. lpr reads from the standard input if no files are specified.

OPTIONS

-P *printer* — Send output to the named *printer*. Otherwise send output to the printer named in the PRINTER environment variable, or to the default printer, lp.

-# *copies* — Produce the number of *copies* indicated for each named file. For example:

```
lpr -#3 index.c lookup.c
```

produces three copies of index.c, followed by three copies of lookup.c. On the other hand,

```
cat index.c lookup.c | lpr -#3
```

generates three copies of the concatenation of the files.

-C *class* — Print *class* as the job classification on the burst page. For example,

```
lpr -C Operations new.index.c
```

replaces the system name (the name returned by *hostname*) with Operations on the burst page, and prints the file new.index.c.

-J *job* — Print *job* as the job name on the burst page. Normally, lpr uses the first file's name.

-T *title* — Use *title* instead of the file name for the title used by pr(1).

-i[*indent*] — Indent output *indent* SPACE characters. Eight SPACE characters is the default.

-w *cols* — Use *cols* as the page width for pr.

-r — Remove the file upon completion of spooling, or upon completion of printing with the -s option. This is not supported in the SunOS compatibility package. However if the job is submitted to a remote SunOS system, these options will be sent to the remote system for processing.

`-m` Send mail upon completion.

`-h` Suppress printing the burst page.

`-s` Use the full pathnames (not symbolic links) of the files to be printed rather than trying to copy them. This means the data files should not be modified or removed until they have been printed. `-s` only prevents copies of local files from being made. Jobs from remote hosts are copied anyway. `-s` only works with named data files; if the `lpr` command is at the end of a pipeline, the data is copied to the spool.

filter_option The following single letter options notify the line printer spooler that the files are not standard text files. The spooling daemon will use the appropriate filters to print the data accordingly.

- `-p` Use `pr` to format the files (`lpr -p` is very much like `pr | lpr`).
- `-l` Print control characters and suppress page breaks.
- `-t` The files contain `troff`(1) (cat phototypesetter) binary data.
- `-n` The files contain data from `ditroff` (device independent troff).
- `-d` The files contain data from `tex` (DVI format from Stanford).
- `-g` The files contain standard plot data as produced by the `plot`(3X) routines (see also `plot`(1G) for the filters used by the printer spooler).
- `-v` The files contain a raster image. The printer must support an appropriate imaging model such as PostScript® in order to print the image.
- `-c` The files contain data produced by *cifplot*.
- `-f` Interpret the first character of each line as a standard FORTRAN carriage control character.

If no *filter_option* is given (and the printer can interpret PostScript), the string '`%!`' as the first two characters of a file indicates that it contains PostScript commands.

These filter options offer a standard user interface, and all options may not be available for, nor applicable to, all printers.

FILES

`/etc/passwd`	personal identification
`/usr/lib/lp/lpsched`	System V line printer spooler
`/var/spool/lp/tmp/*`	directories used for spooling
`/var/spool/lp/tmp/`*system*`/*-0`	spooler control files
`/var/spool/lp/tmp/`*system*`/*-`*N*	(*N* is an integer and > 0) data files specified in '`*-0`' files

DIAGNOSTICS

lpr: *printer* **: unknown printer**
The printer was not found in the LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use 'lptstat -p' to find the reason.

lpr: error on opening queue to spooler
The connection to lpsched on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon /usr/lib/lpsched is running.

lpr: *printer* **: printer queue is disabled**
This means the queue was turned off with

/usr/etc/lpc disable *printer*

to prevent lpr from putting files in the queue. This is normally done by the system manager when a printer is going to be down for a long time. The printer can be turned back on by a privileged user with lpc.

lpr: Can't send message to the LP print service

lpr: Can't receive message from the LP print service
These indicate that the LP print service has been stopped. Get help from the system administrator.

lpr: Received unexpected message from LP print service
It is likely there is an error in this software. Get help from system administrator.

lpr: There is no filter to convert the file content
Use the 'lpstat -p -l' command to find a printer that can handle the file type directly, or consult with your system administrator.

lpr: cannot access the file
Make sure file names are valid.

SEE ALSO

lpc(8), lpq(1), lprm(1), plot(1G), troff(1)

plot(3X) in the *Programmer's Reference Manual.*
lpsched(1) in the *System Administrator's Reference Manual.*
lp(1), pr(1) in the *User's Reference Manual.*

NOTES

lp is the preferred interface.

Command-line options cannot be combined into a single argument as with some other commands. The command:

```
lpr -fs
```

is not equivalent to

```
lpr -f -s
```

Placing the `-s` flag first, or writing each option as a separate argument, makes a link as expected.

`lpr -p` is not precisely equivalent to `pr | lpr`. `lpr -p` puts the current date at the top of each page, rather than the date last modified.

Fonts for `troff`(1) and T_EX® reside on the printer host. It is currently not possible to use local font libraries.

`lpr` objects to printing binary files.

The `-s` option, intended to use symbolic links in SunOS, does not use symbolic links in the compatibility package. Instead, the complete path names are used. Also, the copying is avoided only for print jobs that are run from the printer host itself. Jobs added to the queue from a remote host are always copied into the spool area. That is, if the printer does not reside on the host that `lpr` is run from, the spooling system makes a copy the file to print, and places it in the spool area of the printer host, regardless of `-s`.

NAME

lprm – remove jobs from the printer queue

SYNOPSIS

/usr/ucb/lprm [-P*printer*] [-] [*job #* ...] [*username* ...]

DESCRIPTION

lprm removes a job or jobs from a printer's spooling queue. Since the spool directory is protected from users, using lprm is normally the only method by which a user can remove a job.

Without any arguments, lprm deletes the job that is currently active, provided that the user who invoked lprm owns that job.

When the privileged user specifies a *username*, lprm removes all jobs belonging to that user.

You can remove a specific job by supplying its job number as an argument, which you can obtain using lpq(1). For example:

```
lpq  -Phost
host is ready and printing
Rank Owner    Job     Files    Total Size
active        wendy   385      standard input  35501 bytes
lprm -Phost 385
```

lprm reports the names of any files it removes, and is silent if there are no applicable jobs to remove.

lprm Sends the request to cancel a job to the print spooler, LPSCHED.

OPTIONS

-P*printer* Specify the queue associated with a specific printer. Otherwise the value of the PRINTER variable in the environment is used. If this variable is unset, the queue for the default printer is used.

- Remove all jobs owned by you. If invoked by the privileged user, all jobs in the spool are removed. Job ownership is determined by the user's login name and host name on the machine where the lpr command was executed.

FILES

/var/spool/lp/* spooling directories

SEE ALSO

lpq(1), lpr(1)

lpsched(1M) in the *System Administrator's Reference Manual.*
cancel(1), lp(1) in the *User's Reference Manual.*

DIAGNOSTICS

lprm: *printer*: unknown printer
The printer was not found in the System V LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use 'lptstat -p' to find the reason.

`lprm: error on opening queue to spooler`
The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon `/usr/lib/lpsched` is running.

`lprm: Can't send message to the LP print service`

`lprm: Can't receive message from the LP print service`
These indicate that the LP print service has been stopped. Get help from the system administrator.

`lprm: Received unexpected message from the LP print service`
It is likely there is an error in this software. Get help from system administrator.

`lprm: Can't cancel request`
You are not allowed to remove another's request.

NOTES

An active job may be incorrectly identified for removal by an `lprm` command issued with no arguments. During the interval between an `lpq`(1) command and the execution of `lprm`, the next job in queue may have become active; that job may be removed unintentionally if it is owned by you. To avoid this, supply `lprm` with the job number to remove when a critical job that you own is next in line.

Only the privileged user can remove print jobs submitted from another host.

`lp` is the preferred interface.

NAME

`lptest` – generate lineprinter ripple pattern

SYNOPSIS

`/usr/ucb/lptest` [*length* [*count*]]

DESCRIPTION

`lptest` writes the traditional "ripple test" pattern on standard output. In 96 lines, this pattern will print all 96 printable ASCII characters in each position. While originally created to test printers, it is quite useful for testing terminals, driving terminal ports for debugging purposes, or any other task where a quick supply of random data is needed.

The *length* argument specifies the output line length if the the default length of 79 is inappropriate.

The *count* argument specifies the number of output lines to be generated if the default count of 200 is inappropriate.

NOTES

if *count* is to be specified, *length* must be also be specified.

This command is obsolescent.

NAME

ls – list the contents of a directory

SYNOPSIS

/usr/ucb/ls [-aAcCdfFgilLqrRstu1] *filename* ...

DESCRIPTION

For each *filename* which is a directory, ls lists the contents of the directory; for each *filename* which is a file, ls repeats its name and any other information requested. By default, the output is sorted alphabetically. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments are processed before directories and their contents.

Permissions Field

The mode printed under the -l option contains 10 characters interpreted as follows. If the first character is:

- d entry is a directory;
- b entry is a block-type special file;
- c entry is a character-type special file;
- l entry is a symbolic link;
- p entry is a FIFO (also known as named pipe) special file;
- s entry is an AF_UNIX address family socket, or
- – entry is a plain file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next refers to permissions to others in the same user-group; and the last refers to all others. Within each set the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, execute permission is interpreted to mean permission to search the directory. The permissions are indicated as follows:

- r the file is readable;
- w the file is writable;
- x the file is executable;
- – the indicated permission is not granted.

The group-execute permission character is given as s if the file has the set-group-id bit set; likewise the owner-execute permission character is given as s if the file has the set-user-id bit set.

The last character of the mode (normally x or '–') is true if the 1000 bit of the mode is on. See chmod(1) for the meaning of this mode. The indications of set-ID and 1000 bits of the mode are capitalized (S and T respectively) if the corresponding execute permission is *not* set.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks is printed. The following options are available:

-a List all entries; in the absence of this option, entries whose names begin with a '.' are *not* listed (except for the privileged user, for whom ls normally prints even files that begin with a '.').

-A Same as -a, except that '.' and '. .' are not listed.

-c Use time of last edit (or last mode change) for sorting or printing.

-C Force multi-column output, with entries sorted down the columns; for ls, this is the default when output is to a terminal.

-d If argument is a directory, list only its name (not its contents); often used with -l to get the status of a directory.

-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.

-F Mark directories with a trailing slash ('/'), executable files with a trailing asterisk ('*'), symbolic links with a trailing at-sign ('@'), and AF_UNIX address family sockets with a trailing equals sign ('=').

-g For ls, show the group ownership of the file in a long output.

-i For each file, print the i-node number in the first column of the report.

-l List in long format, giving mode, number of links, owner, size in bytes, and time of last modification for each file. If the file is a special file the size field will instead contain the major and minor device numbers. If the time of last modification is greater than six months ago, it is shown in the format *'month date year'*; files modified within six months show *'month date time'*. If the file is a symbolic link the pathname of the linked-to file is printed preceded by '->'.

-L If argument is a symbolic link, list the file or directory the link references rather than the link itself.

-q Display non-graphic characters in filenames as the character ?; for ls, this is the default when output is to a terminal.

-r Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.

-R Recursively list subdirectories encountered.

-s Give size of each file, including any indirect blocks used to map the file, in kilobytes.

-t Sort by time modified (latest first) instead of by name.

-u Use time of last access instead of last modification for sorting (with the -t option) and/or printing (with the -l option).

-1 Force one entry per line output format; this is the default when output is not to a terminal.

FILES

`/etc/passwd`	to get user ID's for '`ls -l`' and '`ls -o`'.
`/etc/group`	to get group ID for '`ls -g`'

NOTES

NEWLINE and TAB are considered printing characters in filenames.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as '`ls -s`' is much different than '`ls -s | lpr`'. On the other hand, not doing this setting would make old shell scripts which used `ls` almost certain losers.

Unprintable characters in file names may confuse the columnar output options.

NAME

mach – display the processor type of the current host

SYNOPSIS

/usr/ucb/mach

DESCRIPTION

The mach command displays the processor-type of the current host.

SEE ALSO

arch(1).

machid(1), uname(1) in th *User's Reference Manual.*
uname(2), sysinfo(2) in the *Programmer's Reference Manual.*

NAME

man – display reference manual pages; find reference pages by keyword

SYNOPSIS

/usr/ucb/man [-] [-t] [-M *path*] [-T *macro-package*] [[*section*] *title*...]
title...
/usr/ucb/man [-M *path*] -k *keyword* ...
/usr/ucb/man [-M *path*] -f *filename* ...

DESCRIPTION

The man command displays information from the reference manuals. It can display complete manual pages that you select by *title*, or one-line summaries selected either by *keyword* (-k), or by the name of an associated file (-f).

A *section*, when given, applies to the *titles* that follow it on the command line (up to the next *section*, if any). man looks in the indicated section of the manual for those *titles*. *section* is either a digit (perhaps followed by a single letter indicating the type of manual page), or one of the words **new**, local, old, or public. If *section* is omitted, man searches all reference sections (giving preference to commands over functions) and prints the first manual page it finds. If no manual page is located, man prints an error message.

The reference page sources are typically located in the /usr/share/man/man? directories. Since these directories are optionally installed, they may not reside on your host; you may have to mount /usr/share/man from a host on which they do reside. If there are preformatted, up-to-date versions in corresponding cat? or fmt? directories, man simply displays or prints those versions. If the preformatted version of interest is out of date or missing, man reformats it prior to display. If directories for the preformatted versions are not provided, man reformats a page whenever it is requested; it uses a temporary file to store the formatted text during display.

If the standard output is not a terminal, or if the - flag is given, man pipes its output through cat. Otherwise, man pipes its output through more to handle paging and underlining on the screen.

The following options are available:

-t man arranges for the specified manual pages to be troffed to a suitable raster output device (see troff or vtroff). If both the - and -t flags are given, man updates the troffed versions of each named *title* (if necessary), but does not display them.

-M *path*
Change the search path for manual pages. *path* is a colon-separated list of directories that contain manual page directory subtrees. When used with the -k or -f options, the -M option must appear first. Each directory in the *path* is assumed to contain subdirectories of the form man[1-8l-p].

-T *macro-package*
man uses *macro-package* rather than the standard -man macros defined in /usr/ucblib/doctools/tmac/tmac.an for formatting manual pages.

-k *keyword ...*
man prints out one-line summaries from the `whatis` database (table of contents) that contain any of the given *keyword*s.

-f *filename ...*
man attempts to locate manual pages related to any of the given *filenames*. It strips the leading pathname components from each *filename*, and then prints one-line summaries containing the resulting basename or names.

MANUAL PAGES

Manual pages are `troff` or `nroff` source files prepared with the -man macro package.

When formatting a manual page, man examines the first line to determine whether it requires special processing.

Preprocessing Manual Pages

If the first line is a string of the form:

```
'\" X
```

where X is separated from the the '"' by a single SPACE and consists of any combination of characters in the following list, man pipes its input to `troff` or `nroff` through the corresponding preprocessors.

`e`	eqn, or neqn for `nroff`
`r`	`refer`
`t`	`tbl`, and `col` for `nroff`

If `eqn` or `neqn` is invoked, it will automatically read the file `/usr/ucblib/pub/eqnchar` [see `eqnchar`(7)].

ENVIRONMENT

`MANPATH`	If set, its value overrides `/usr/share/man` as the default search path. The `-M` flag, in turn, overrides this value.
`PAGER`	A program to use for interactively delivering man's output to the screen. If not set, '`more -s`' (see `more`) is used.
`TCAT`	The name of the program to use to display `troff`ed manual pages. If not set, '`lp -Ttroff`' (see `lp`) is used.
`TROFF`	The name of the formatter to use when the `-t` flag is given. If not set, `troff` is used.

FILES

`/usr/share/man`	root of the standard manual page directory subtree
`/usr/share/man/man?/*`	unformatted manual entries
`/usr/share/man/cat?/*`	`nroff`ed manual entries
`/usr/share/man/fmt?/*`	`troff`ed manual entries
`/usr/share/man/whatis`	table of contents and keyword database
`/usr/ucblib/doctools/tmac/man.macs`	standard -man macro package
`/usr/ucblib/pub/eqnchar`	

SEE ALSO

apropos(1), cat(1), catman(1M), col(1), eqn(1), nroff(1), refer(1), tbl(1), troff(1), whatis(1), eqnchar(7).

col(1), lp(1), more(1) in the *User's Reference Manual.*

NOTES

The manual is supposed to be reproducible either on a phototypesetter or on an ASCII terminal. However, on a terminal some information (indicated by font changes, for instance) is necessarily lost.

Some dumb terminals cannot process the vertical motions produced by the e (eqn(1)) preprocessing flag. To prevent garbled output on these terminals, when you use e also use t, to invoke col(1) implicitly. This workaround has the disadvantage of eliminating superscripts and subscripts — even on those terminals that can display them. CTRL-Q will clear a terminal that gets confused by eqn(1) output.

NAME

mt – magnetic tape control

SYNOPSIS

/usr/ucb/mt [-f *tapename*] *command* [*count*]

DESCRIPTION

mt sends commands to a magnetic tape drive. If *tapename* is not specified, the environment variable TAPE is used. If TAPE does not exist, mt uses the device /dev/rmt12. *tapename* must refer to a raw (not block) tape device. By default, mt performs the requested operation once; multiple operations may be performed by specifying *count*.

The available commands are listed below. Only as many characters as are required to uniquely identify a command need be specified.

mt returns a 0 exit status when the operation(s) were successful, 1 if the command was unrecognized or if mt was unable to open the specified tape drive, and 2 if an operation failed.

the following commands are available to mt:

eof, weof	Write *count* EOF marks at the current position on the tape.
fsf	Forward space *count* files.
fsr	Forward space *count* records.
bsf	Back space *count* files.
bsr	Back space *count* records.
asf	Absolute space to *count* file number. This is equivalent to a rewind followed by a fsf *count*.

For the following commands, *count* is ignored:

eom	Space to the end of recorded media on the tape (SCSI only). This is useful for appending files onto previously written tapes.
rewind	Rewind the tape.
offline, rewoffl	Rewind, unload, and place the tape drive unit off-line.
status	Print status information about the tape unit.
retension	Wind the tape to the end of the reel and then rewind it, smoothing out the tape tension.
erase	Erase the entire tape.

FILES

/dev/rmt*	raw magnetic tape interface
/dev/rar*	raw Archive cartridge tape interface
/dev/rst*	raw SCSI tape interface
/dev/rmt*	raw Xylogics® tape interface

SEE ALSO

dd(1M), ar(4), environ(5), xt(7) in the *System Administrator's Reference Manual.*

NOTES

Not all devices support all options. For example, `ar` currently does not support the `fsr`, `bsf`, or `bsr` options. The half-inch tape driver, `/dev/rmt*`, does not support the `retension` option.

NAME

nroff – format documents for display or line-printer

SYNOPSIS

/usr/ucb/nroff [-ehiqz] [-F*dir*] [-m*name*] [-n*N*] [-o*pagelist*] [-r*aN*] [-s*N*] [-T*name*] [-u*N*] [*filename* ...]

DESCRIPTION

nroff formats text in the named *filename* for typewriter-like devices. See also troff.

If no *filename* argument is present, nroff reads the standard input. An argument consisting of a '–' is taken to be a file name corresponding to the standard input.

The following options may appear in any order, but must appear *before* the files.

-e Produce equally-spaced words in adjusted lines, using full terminal resolution.

-h Use output TAB characters during horizontal spacing to speed output and reduce output character count. TAB settings are assumed to be every 8 nominal character widths.

-i Read the standard input after the input files are exhausted.

-q Invoke the simultaneous input-output mode of the rd request.

-F*dir* Search directory *dir* for font tables instead of the system-dependent default.

-m*name*
Prepend the macro file /usr/share/lib/tmac/tmac.*name* to the input files.

-n*N* Number first generated page *N* .

-o*pagelist*
Print only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N*– M means pages *N* through *M*; an initial –*N* means from the beginning to page *N*; and a final *N*– means from *N* to the end.

-r*aN* Set register *a* (one-character) to *N*.

-s*N* Stop every *N* pages. nroff will halt prior to every *N* pages (default *N*=1) to allow paper loading or changing, and will resume upon receipt of a NEWLINE.

-T*name*
Prepare output for a device of the specified name. Known names are:

37 Teletype Corporation Model 37 terminal — this is the default.

crt | lpr | tn300
GE TermiNet 300, or any line printer or terminal without half-line capability.

300 DASI-300.

300-12 DASI-300 — 12-pitch.
300S | 302 | dtc
DASI-300S.
300S-12 | 302-12 | dtc12
DASI-300S.
382 DASI-382 (fancy DTC 382).
382-12 DASI-82 (fancy DTC 382 — 12-pitch).
450 | ipsi
DASI-450 (Diablo Hyterm).
450-12 | ipsi12
DASI-450 (Diablo Hyterm) — 12-pitch.
450-12-8 DASI-450 (Diablo Hyterm) — 12-pitch and 8 lines-per-inch.
450X DASI-450X (Diablo Hyterm).
832 AJ 832.
833 AJ 833.
832-12 AJ 832 — 12-pitch.
833-12 AJ 833 — 12-pitch.
epson Epson FX80.
itoh C:ITOH Prowriter.
itoh-12 C:ITOH Prowriter — 12-pitch.
nec NEC 55?0s0 or NEC 77?0s0 Spinwriter.
nec12 NEC 55?0 or NEC 77?0 Spinwriter — 12-pitch.
nec-t NEC 55?0/77?0 Spinwriter — Tech-Math/Times-Roman thimble.
qume Qume Sprint — 5 or9.
qume12 Qume Sprint — 5 or 9,12-pitch.
xerox Xerox 17?0 or Diablo 16?0.
xerox12 Xerox 17?0 or Diablo 16?0 — 12-pitch.
x-ecs Xerox/Diablo 1730/630 — Extended Character Set.
x-ecs12 Xerox/Diablo 1730/630 — Extended Character Set, 12-pitch.

-u*N* Set emboldening factor for the font mounted on position 3 to N. Emboldening is accomplished by overstriking the specified number of times.

-z Suppress formatted output. The only output will consist of diagnostic messages from nroff and messages output with the .tm request.

EXAMPLE

The following command:

```
nroff -s4 -me users.guide
```

formats users.guide using the -me macro package, and stopping every 4 pages.

FILES

/tmp/ta*	temporary file
/usr/ucblib/doctools/tmac/tmac.*	standard macro files
/usr/ucblib/doctools/term/*	terminal driving tables for nroff
/usr/ucblib/doctools/term/README	index to terminal description files

SEE ALSO

checknr(1), eqn(1), tbl(1), troff(1), man(7), me(7), ms(7).

col(1) in the *User's Reference Manual.*
term(4) in the *System Administrator's Reference Manual.*

NAME

pagesize – display the size of a page of memory

SYNOPSIS

```
/usr/ucb/pagesize
```

DESCRIPTION

pagesize prints the size of a page of memory in bytes, as returned by getpagesize. This program is useful in constructing portable shell scripts.

SEE ALSO

getpagesize(3)

NAME

plot, aedplot, bgplot, crtplot, dumbplot, gigiplot, hpplot, implot, t300, t300s, t4013, t450, tek – graphics filters for various plotters

SYNOPSIS

/usr/ucb/plot [-T*terminal*]

DESCRIPTION

plot reads plotting instructions [see plot(4)] from the standard input and produces plotting instructions suitable for a particular *terminal* on the standard output.

If no *terminal* is specified, the environment variable TERM is used. The default *terminal* is tek.

ENVIRONMENT

Except for ver, the following terminal-types can be used with 'lpr -g' (see lpr) to produce plotted output:

2648 | 2648a | h8 | hp2648 | hp2648a
: Hewlett Packard® 2648 graphics terminal.

300
: DASI 300 or GSI terminal (Diablo® mechanism).

300s | 300S
: DASI 300s terminal (Diablo mechanism).

450
: DASI Hyterm 450 terminal (Diablo mechanism).

4013
: Tektronix® 4013 storage scope.

4014 | tek
: Tektronix 4014 and 4015 storage scope with Enhanced Graphics Module. (Use 4013 for Tektronix 4014 or 4015 without the Enhanced Graphics Module).

aed
: AED 512 color graphics terminal.

bgplot | bitgraph
: BBN bitgraph graphics terminal.

crt
: Any crt terminal capable of running vi(1).

dumb | un | unknown
: Dumb terminals without cursor addressing or line printers.

gigi | vt125
: DEC® vt125 terminal.

h7 | hp7 | hp7221
: Hewlett Packard 7221 graphics terminal.

implot
: Imagen plotter.

var
: Benson Varian printer-plotter

ver
: Versatec® D1200A printer-plotter. The output is scan-converted and suitable input to 'lpr -v'.

FILES

/usr/ucb/aedplot
/usr/ucb/bgplot
/usr/ucb/crtplot
/usr/ucb/dumbplot
/usr/ucb/gigiplot
/usr/ucb/hpplot
/usr/ucb/implot
/usr/ucb/plot
/usr/ucb/t300
/usr/ucb/t300s
/usr/ucb/t4013
/usr/ucb/t450
/usr/ucb/tek
/usr/ucb/vplot
/var/ucb/vplot*nnnnnn*

SEE ALSO

lpr(1).

vi(1) in the *User's Reference Manual.*
plot(3X), plot(4) in the *Programmer's Reference Manual.*

NAME

printenv – display environment variables currently set

SYNOPSIS

/usr/ucb/printenv [*variable*]

DESCRIPTION

printenv prints out the values of the variables in the environment. If a *variable* is specified, only its value is printed.

SEE ALSO

tset(1).

csh(1), echo(1), sh(1), stty(1) in the *User's Reference Manual.*
environ(5) in the *System Administrator's Reference Manual.*

DIAGNOSTICS

If a *variable* is specified and it is not defined in the environment, printenv returns an exit status of 1.

NAME

prt – display the delta and commentary history of an SCCS file

SYNOPSIS

/usr/ucb/prt [-abdefistu] [-y[*SID*]] [-c[*cutoff*]] [-r[*rev-cutoff*]] *filename...*

DESCRIPTION

Note: the prt command is an older version of prs(1) that in most circumstances is more convenient to use, but is less flexible than prs.

prt prints part or all of an SCCS file in a useful format. If a directory is named, prt behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the pathname does not begin with s.) and unreadable files are silently ignored. If a name of '–' is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

The following options are available with prt :

-a Print those types of deltas normally not printed by the d keyletter. These are types R (removed). This keyletter is effective only if the d keyletter is also specified (or assumed).

-b Print the body of the SCCS file.

-d This keyletter normally prints delta table entries of the D type.

-e This keyletter implies the d, i, u, f, and t keyletters and is provided for convenience.

-f Print the flags of the named file.

-i Print the serial numbers of those deltas included, excluded, and ignored. This keyletter is effective only if the d keyletter is also specified (or assumed).

The following format is used to print those portions of the SCCS file as specified by the above keyletters. The printing of each delta table entry is preceded by a NEWLINE.

- Type of delta (D or R).
- SPACE.
- SCCS identification string (SID).
- TAB.
- Date and time of creation (in the form YY/MM/DD HH:MM:SS).
- SPACE.
- Creator.
- TAB.
- Serial number.
- SPACE.
- Predecessor delta's serial number.
- TAB.

- Statistics (in the form inserted/deleted/unchanged).
- NEWLINE.
- "Included:TAB", followed by SID's of deltas included, followed by NEWLINE (only if there were any such deltas and if `i` keyletter was supplied).
- "Excluded:TAB", followed by SID's of deltas excluded, followed by NEWLINE (see note above).
- "Ignored:TAB", followed by SID's of deltas ignored, followed by NEWLINE (see note above).
- "MRs:TAB", followed by MR numbers related to the delta, followed by NEWLINE (only if any MR numbers were supplied).
- Lines of comments (delta commentary), followed by newline (if any were supplied).

`-s` Print only the first line of the delta table entries; that is, only up to the statistics. This keyletter is effective only if the `d` keyletter is also specified (or assumed).

`-t` Print the descriptive text contained in the file.

`-u` Print the login-names and/or numerical group IDs of those users allowed to make deltas.

`-y`[*SID*]
Print the delta table entries to stop when the delta just printed has the specified SID. If no delta in the table has the specified SID, the entire table is printed. If no SID is specified, the first delta in the delta table is printed. This keyletter will print the entire delta table entry for each delta as a single line (the NEWLINE in the normal multi-line format of the `d` keyletter are replaced by SPACE characters) preceded by the name of the SCCS file being processed, followed by a `:`, followed by a TAB. This keyletter is effective only if the `d` keyletter is also specified (or assumed).

`-c`[*cutoff*]
Stop printing the delta table entries if the delta about to be printed is older than the specified cutoff date-time (see `get`(1) for the format of date-time). If no date-time is supplied, the epoch 0000 GMT Jan. 1, 1970 is used. As with the `y` keyletter, this keyletter will cause the entire delta table entry to be printed as a single line and to be preceded by the name of the SCCS file being processed, followed by a `:`, followed by a tab. This keyletter is effective only if the `d` keyletter is also specified (or assumed).

`-r`[*rev-cutoff*]
Begin printing the delta table entries when the delta about to be printed is older than or equal to the specified cutoff date-time (see `get`(1) for the format of date-time). If no date-time is supplied, the epoch 0000 GMT Jan. 1, 1970 is used. (In this case, nothing will be printed). As with the `y` keyletter, this keyletter will cause the entire delta table entry to be printed as a single line and to be preceded by the name of the SCCS file being processed, followed by a `:`, followed by a tab. This keyletter is effective only if the `d` keyletter is also specified (or assumed).

If any keyletter but `y`, `c`, or `r` is supplied, the name of the file being processed (preceded by one NEWLINE and followed by two NEWLINE characters) is printed before its contents.

If none of the `u`, `f`, `t`, or `b` keyletters is supplied, the `d` keyletter is assumed.

Note: the `s` and `i` keyletters, and the `c` and `r` keyletters are mutually exclusive; therefore, they may not be specified together on the same `prt` command.

The form of the delta table as produced by the `y`, `c`, and `r` keyletters makes it easy to sort multiple delta tables in chronological order.

When both the `y` and `c` or the `y` and `r` keyletters are supplied, `prt` will stop printing when the first of the two conditions is met.

SEE ALSO

`admin`(1), `get`(1), `delta`(1), `prs`(1), `what`(1), `sccs`(1).

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

NAME

ps – display the status of current processes

SYNOPSIS

/usr/ucb/ps [-acglnrSuUvwx] [-t*term*] [*num*]

DESCRIPTION

The ps command displays information about processes. Normally, only those processes that are running with your effective user ID and are attached to a controlling terminal (see termio(4)) are shown. Additional categories of processes can be added to the display using various options. In particular, the -a option allows you to include processes that are not owned by you (that do not have your user ID), and the -x option allows you to include processes without control terminals. When you specify both -a and -x, you get processes owned by anyone, with or without a control terminal. The -r option restricts the list of processes printed to running and runnable processes.

ps displays the process ID, under PID; the control terminal (if any), under TT; the cpu time used by the process so far, including both user and system time, under TIME; the state of the process, under S; and finally, an indication of the COMMAND that is running.

The state is given by a single letter from the following:

O	Process is running on a processor.
S	Sleeping. Process is waiting for an event to complete.
R	Runnable. Process is on run queue.
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.

The following options must all be combined to form the first argument:

-a Include information about processes owned by others.

-c Display the command name, as stored internally in the system for purposes of accounting, rather than the command arguments, which are kept in the process' address space. This is more reliable, if less informative, since the process is free to destroy the latter information.

-g Display all processes. Without this option, ps only prints interesting processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to login on free terminals.

-l Display a long listing, with fields F, PPID, CP, PRI, NI, SZ, RSS and WCHAN as described below.

-n Produce numerical output for some fields. In a user listing, the USER field is replaced by a UID field.

-r Restrict output to running and runnable processes.

-S Display accumulated CPU time used by this process and all of its reaped children.

-u Display user-oriented output. This includes fields USER, SZ, RSS and START as described below.

-U Update a private database where ps keeps system information.

-v Display a version of the output containing virtual memory. This includes fields SIZE and RSS, described below.

-w Use a wide output format (132 columns rather than 80); if repeated, that is, -ww, use arbitrarily wide output. This information is used to decide how much of long commands to print.

-x Include processes with no controlling terminal.

-t*term*
List only process data associated with the terminal, *term*. Terminal identifiers may be specified in one of two forms: the device's file name (for example, tty04 or term/14) or, if the device's file name starts with tty, just the digit identifier (for example, 04).

num A process number may be given, in which case the output is restricted to that process. This option must be supplied last.

DISPLAY FORMATS

Fields that are not common to all output formats:

USER Name of the owner of the process.

NI Process scheduling increment [see getpriority(3) and nice(3C)].

SIZE
SZ The combined size of the data and stack segments (in kilobyte units)

RSS Real memory (resident set) size of the process (in kilobyte units).

UID Numerical user-ID of process owner.

PPID Numerical ID of parent of process.

CP Short-term CPU utilization factor (used in scheduling).

PRI The priority of the process (higher numbers mean lower priority).

START The starting time of the process, given in hours, minutes, and seconds. A process begun more than 24 hours before the ps inquiry is executed is given in months and days.

WCHAN The address of an event for which the process is sleeping, or in SXBRK state (if blank, the process is running).

F Flags (hexadecimal and additive) associated with the process:
00 Process has terminated. Process table 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, see ptrace(2).

08 Process is currently in primary memory.
10 Process currently in primary memory, locked until an event is completed.

A process that has exited and has a parent, but has not yet been waited for by the parent is marked <defunct>; otherwise, ps tries to determine the command name and arguments given when the process was created by examining the user block.

FILES

/dev	
/dev/sxt/*	
/dev/tty*	
/dev/xt/*	terminal (tty) names searcher files
/dev/kmem	kernel virtual memory
/dev/swap	default swap device
/dev/mem	memory
/etc/passwd	UID information supplier
/etc/ps_data	internal data structure

SEE ALSO

getpriority(3), nice(3C)

kill(1) in the *User's Reference Manual.*
whodo(1) in the *System Administrator's Reference Manual.*
lseek(2) in the *Programmer's Reference Manual.*

NOTES

Things can change while ps is running; the picture it gives is only a close approximation to the current state. Some data printed for defunct processes is irrelevant.

If no *term* or *num* is specified, ps checks the standard input, the standard output, and the standard error in that order, looking for the controlling terminal and will attempt to report on processes associated with the controlling terminal. In this situation, if the standard input, the standard output, and the standard error 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 process' user area, ps may not be able to seek to that address before the process exits and the address becomes invalid.

NAME

refer – expand and insert references from a bibliographic database

SYNOPSIS

/usr/ucb/refer [-b] [-e] [-n] [-a*r*] [-c*string*] [-k*x*] [-l*m,n*] [-p *filename*] [-s*keys*] *filename*...

DESCRIPTION

refer is a preprocessor for nroff(1), or troff(1), that finds and formats references. The input files (standard input by default) are copied to the standard output, except for lines between '. [' and '.]' command lines. Such lines are assumed to contain keywords as for lookbib(1), and are replaced by information from a bibliographic data base. The user can avoid the search, override fields from it, or add new fields. The reference data, from whatever source, is assigned to a set of troff strings. Macro packages such as ms(7) print the finished reference text from these strings. A flag is placed in the text at the point of reference. By default, the references are indicated by numbers.

When refer is used with eqn(1), neqn, or tbl(1), refer should be used first in the sequence, to minimize the volume of data passed through pipes.

The following options are available:

-b Bare mode — do not put any flags in text (neither numbers or labels).

-e Accumulate references instead of leaving the references where encountered, until a sequence of the form:

```
.[
$LIST$
.]
```

is encountered, and then write out all references collected so far. Collapse references to the same source.

-n Do not search the default file.

-a*r* Reverse the first *r* author names (Jones, J. A. instead of J. A. Jones). If *r* is omitted, all author names are reversed.

-c*string*
Capitalize (with SMALL CAPS) the fields whose key-letters are in *string*.

-k*x* Instead of numbering references, use key labels as specified in a reference data line beginning with the characters %*x*; By default, %*x* is %L.

-l*m,n* Instead of numbering references, use labels from the senior author's last name and the year of publication. Only the first *m* letters of the last name and the last *n* digits of the date are used. If either of *m* or *n* is omitted, the entire name or date, respectively, is used.

-p *filename*
Take the next argument as a file of references to be searched. The default file is searched last.

-s*keys* Sort references by fields whose key-letters are in the *keys* string, and permute reference numbers in the text accordingly. Using this option implies the -e option. The key-letters in *keys* may be followed by a number indicating how many such fields are used, with a + sign taken as a very large number. The default is AD, which sorts on the senior author and date. To sort on all authors and then the date, for instance, use the options '-sA+T'.

FILES

/usr/ucblib/reftools/papers	default publication lists and indexes
/usr/ucblib/reftools	programs

SEE ALSO

addbib(1), eqn(1), indxbib(1), lookbib(1), nroff(1), roffbib(1), sortbib(1), tbl(1), troff(1).

NAME

roffbib – format and print a bibliographic database

SYNOPSIS

/usr/ucb/roffbib [-e] [-h] [-Q] [-x] [-m *filename*] [-n*p*] [-o*list*]
[-r*aN*] [-s*N*] [-T*term*] [*filename*] ...

DESCRIPTION

The roffbib command prints out all records in a bibliographic database, in bibliography format rather than as footnotes or endnotes. Generally it is used in conjunction with sortbib(1):

```
example% sortbib  database | roffbib
```

If abstracts or comments are entered following the %X field key, roffbib will format them into paragraphs for an annotated bibliography. Several %X fields may be given if several annotation paragraphs are desired.

roffbib accepts all options understood by nroff(1) except -i and -q, as well as those listed below:

-e Produce equally-spaced words in adjusted lines using full terminal resolution.

-h Use output tabs during horizontal spacing to speed output and reduce output character count. TAB settings are assumed to be every 8 nominal character widths.

-Q Queue output for the phototypesetter. Page offset is set to 1 inch.

-x Suppress printing of abstracts.

-m *filename*
Prepend the macro file /usr/ucblib/doctools/tmac/tmac.name to the input files. There should be a space between the -m and the macro filename. This set of macros will replace the ones defined in /usr/ucblib/doctools/tmac/tmac.bib.

-n*p* Number first generated page *p*.

-o*list* Print only page numbers that appear in the comma-separated *list* of numbers and ranges. A range *N*–*M* means pages *N* through *M*; an initial –*N* means from the beginning to page *N*; a final *N*– means from page *N* to end.

-r*aN* Set register *a* (one-character) to *N*. The command-line argument -rN1 will number the references starting at 1.

Four command-line registers control formatting style of the bibliography, much like the number registers of ms(7). The flag -rV2 will double space the bibliography, while -rV1 will double space references but single space annotation paragraphs. The line length can be changed from the default 6.5 inches to 6 inches with the -rL6i argument, and the page offset can be set from the default of 0 to one inch by specifying -rO1i (capital O, not zero).

-s*N* Halt prior to every *N* pages for paper loading or changing (default *N*=1). To resume, enter NEWLINE or RETURN.

-T*term*
Specify *term* as the terminal type.

FILES

`/usr/ucblib/doctools/tmac/tmac.bib`
file of macros used by `nroff/troff`

SEE ALSO

`addbib`(1), `indxbib`(1), `lookbib`(1), `nroff`(1) `refer`(1), `sortbib`(1), `troff`(1).

NOTES

Users have to rewrite macros to create customized formats.

NAME

sccs – front end for the Source Code Control System (SCCS)

SYNOPSIS

/usr/ucb/sccs [-r] [-d*prefixpath*] [-p*finalpath*] *command* [*SCCS-flags* ...] [*filename* ...]

DESCRIPTION

The sccs command is a front end to the utility programs of the Source Code Control System (SCCS).

sccs normally prefixes each *filename*, or the last component of each *filename*, with the string 'SCCS/s.', because you normally keep your SCCS database files in a directory called SCCS, and each database file starts with an 's.' prefix. If the environment variable PROJECTDIR is set, and is an absolute pathname (that is, begins with a slash) sccs will search for SCCS files in the directory given by that variable. If it is a relative pathname (that is, does not begin with a slash), it is treated as the name of a user, and sccs will search in that user's home directory for a directory named src or source. If that directory is found, sccs will search for SCCS files in the directory given by that variable.

sccs program options must appear before the *command* argument. Flags to be passed to the actual SCCS command (utility program) must appear after the *command* argument. These flags are specific to the *command* being used.

sccs also includes the capability to run "set user ID" to another user to provide additional protection. Certain commands (such as admin(1)) cannot be run "set user ID" by all users, since this would allow anyone to change the authorizations. Such commands are always run as the real user.

OPTIONS

-r Run sccs as the real user rather than as whatever effective user sccs is "set user ID" to.

-d*prefixpath*
Define the prefix portion of the pathname for the SCCS database files. The default prefix portion of the pathname is the current directory. *prefixpath* is prefixed to the entire pathname. See EXAMPLE.

This flag overrides any directory specified by the PROJECTDIR environment variable.

-p*finalpath*
Define the name of a lower directory in which the SCCS files will be found; SCCS is the default. *finalpath* is appended before the final component of the pathname. See EXAMPLE.

USAGE

Additional sccs Commands

Several "pseudo-commands" are available in addition to the usual SCCS commands. These are:

create create is used when creating new s. files. For example, given a C source language file called 'obscure.c', create would perform the following actions: (1) create the 's.' file called 's.obscure.c' in the SCCS directory; (2) rename the original source file to ',obscure.c';

(3) do an 'sccs get' on 'obscure.c'. Compared to the SCCS admin command, create does more of the startup work for you and should be used in preference to admin.

enter — enter is just like create, except that it does not do the final 'sccs get'. It is usually used if an 'sccs edit' is to be performed immediately after the enter.

edit — Get a file for editing.

delget — Perform a delta on the named files and then get new versions. The new versions have ID keywords expanded, and so cannot be edited.

deledit — Same as delget, but produces new versions suitable for editing. deledit is useful for making a "checkpoint" of your current editing phase.

fix — Remove the named delta, but leaves you with a copy of the delta with the changes that were in it. fix must be followed by a -r flag. fix is useful for fixing small compiler bugs, etc. Since fix does not leave audit trails, use it carefully.

clean — Remove everything from the current directory that can be recreated from SCCS files. clean checks for and does not remove any files being edited. If 'clean -b' is used, branches are not checked to see if they are currently being edited. Note: -b is dangerous if you are keeping the branches in the same directory.

unedit — "Undo" the last edit or 'get -e' and return a file to its previous condition. If you unedit a file being edited, all changes made since the beginning of the editing session are lost.

info — Display a list of all files being edited. If the -b flag is given, branches (that is, SID's with two or fewer components) are ignored. If the -u flag is given (with an optional argument), only files being edited by you (or the named user) are listed.

check — Check for files currently being edited, like info, but returns an exit code rather than a listing: nothing is printed if nothing is being edited, and a non-zero exit status is returned if anything is being edited. check may thus be included in an "install" entry in a makefile, to ensure that everything is included in an SCCS file before a version is installed.

tell — Display a list of files being edited on the standard output. Filenames are separated by NEWLINE characters. Take the -b and -u flags like info and check.

diffs — Compare (in diff-like format) the current version of the program you have out for editing and the versions in SCCS format. diffs accepts the same arguments as diff, except that the -c flag must be specified as -C instead, because the -c flag is taken as a flag to get indicating which version is to be compared with the current version.

`print` Print verbose information about the named files. `print` does an 'sccs prs -e' followed by an 'sccs get -p -m' on each file.

EXAMPLE

The command:

```
sccs -d/usr/include get sys/inode.h
```

converts to:

```
get /usr/include/sys/SCCS/s.inode.h
```

The intent here is to create aliases such as:

```
alias syssccs sccs -d/usr/src
```

which will be used as:

```
syssccs get cmd/who.c
```

The command:

```
sccs -pprivate get usr/include/stdio.h
```

converts to:

```
get usr/include/private/s.stdio.h
```

To put a file called `myprogram.c` into SCCS format for the first time, assuming also that there is no SCCS directory already existing:

```
$ mkdir SCCS
$ sccs create myprogram.c
$ myprogram.c:
1.1
14 lines
after you have verified that everything is all right
you remove the version of the file that starts with a comma:
$ rm myprogram.c
$
```

To get a copy of `myprogram.c` for editing, edit that file, then place it back in the SCCS database:

```
$ sccs edit myprogram.c
1.1
new delta 1.2
14 lines
$ vi myprogram.c
your editing session
$ sccs delget myprogram.c
comments? Added abusive responses for compatibility
1.2
7 inserted
7 deleted
7 unchanged
1.2
14 lines
$
```

To get a file from another directory:

```
sccs -p/usr/src/sccs/  get cc.c
```

or:

```
sccs get /usr/src/sccs/cc.c
```

To make a delta of a large number of files in the current directory:

```
sccs delta *.c
```

To get a list of files being edited that are not on branches:

```
sccs info -b
```

To delta everything that you are editing:

```
$ sccs delta `sccs tell -u`
```

In a makefile, to get source files from an SCCS file if it does not already exist:

```
SRCS = <list of source files>
$(SRCS):
        sccs get $(REL) $@
```

Regular sccs Commands

The "regular" SCCS commands are described very briefly below. It is unlikely that you ever need to use these commands because the user interface is so complicated, and the sccs front end command does 99.9% of the interesting tasks for you.

admin	Create new SCCS files and changes parameters of existing SCCS files. You can use 'sccs create' to create new SCCS files, or use 'sccs admin' to do other things.
cdc	Change the commentary material in an SCCS delta.
comb	Combine SCCS deltas and reconstructs the SCCS files.
delta	Permanently introduces changes that were made to a file previously retrieved using 'sccs get'. You can use 'sccs delget' as the more useful version of this command since 'sccs delget' does all of the useful work and more.
get	Extract a file from the SCCS database, either for compilation, or for editing when the -e option is used. Use 'sccs get' if you really need it, but 'sccs delget' will normally have done this job for you. Use sccs edit instead of get with the -e option.
help	Supposed to help you interpret SCCS error messages.
prs	Display information about what is happening in an SCCS file.
rmdel	Remove a delta from an SCCS file.
sccsdiff	Compare two versions of an SCCS file and generates the differences between the two versions.

`val` Determine if a given SCCS file meets specified criteria. If you use the `sccs` command, you should not need to use `val`, because its user interface is unbelievable.

`what` Display SCCS identification information.

FILES

`/usr/sccs/*`

SEE ALSO

`admin`(1), `cdc`(1), `comb`(1), `delta`(1), `get`(1), `help`(1), `prs`(1), `rmdel`(1), `sact`(1), `sccsdiff`(1), `unget`(1), `val`(1), `what`(1), `sccsfile`(5) in the *Programmer's Reference Manual.*

NOTES

The `help` command usually just parrots SCCS error messages and is generally not considered very helpful.

NAME

soelim – resolve and eliminate .so requests from nroff or troff input

SYNOPSIS

/usr/ucb/soelim [*filename* ...]

DESCRIPTION

The soelim command reads the specified files or the standard input and performs the textual inclusion implied by the nroff(1) directives of the form

.so *somefile*

when they appear at the beginning of input lines. This is useful since programs such as tbl(1) do not normally do this; it allows the placement of individual tables in separate files to be run as a part of a large document.

An argument consisting of '–' is taken to be a file name corresponding to the standard input.

Note: inclusion can be suppressed by using ' ' ' instead of '.', that is,

```
so /usr/ucblib/doctools/tmac/tmac.s
```

EXAMPLE

A sample usage of soelim would be

```
soelim exum?.n | tbl | nroff -ms | col | lpr
```

SEE ALSO

nroff(1), tbl(1).

more(1) in the *User's Reference Manual.*

NAME

sortbib – sort a bibliographic database

SYNOPSIS

/usr/ucb/sortbib [-s*key-letters*] *database...*

DESCRIPTION

The sortbib command sorts files of records containing refer key-letters by user-specified keys. Records may be separated by blank lines, or by '.[' and '.]' delimiters, but the two styles may not be mixed together. This program reads through each *database* and pulls out key fields, which are sorted separately. The sorted key fields contain the file pointer, byte offset, and length of corresponding records. These records are delivered using disk seeks and reads, so sortbib may not be used in a pipeline to read standard input.

By default, sortbib alphabetizes by the first %A and the %D fields, which contain the senior author and date. The -s option is used to specify new *key-letters*. See addbib for a list of the most common key letters. For instance, -sATD will sort by author, title, and date, while -sA+D will sort by all authors, and date. Sort keys past the fourth are not meaningful. No more than 16 databases may be sorted together at one time. Records longer than 4096 characters will be truncated.

sortbib sorts on the last word on the %A line, which is assumed to be the author's last name. A word in the final position, such as 'jr.' or 'ed.', will be ignored if the name beforehand ends with a comma. Authors with two-word last names or unusual constructions can be sorted correctly by using the nroff convention '\0' in place of a blank. A %Q field is considered to be the same as %A, except sorting begins with the first, not the last, word. sortbib sorts on the last word of the %D line, usually the year. It also ignores leading articles (like 'A' or 'The') when sorting by titles in the %T or %J fields; it will ignore articles of any modern European language. If a sort-significant field is absent from a record, sortbib places that record before other records containing that field.

SEE ALSO

addbib(1), indxbib(1), lookbib(1), refer(1), roffbib(1).

NOTES

Records with missing author fields should probably be sorted by title.

NAME

stty – set the options for a terminal

SYNOPSIS

/usr/ucb/stty [-a] [-g] [-h] [*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.

-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.

-h reports all the option settings with the control characters in an easy to read column format.

Options in the last group are implemented using options in the previous groups. Note that many combinations of options make no sense, but no sanity checking is performed. Hardware flow control and clock modes options may not be supported by all hardware interfaces. The options are selected from the following:

Special Requests

all	Reports the same option settings as stty without arguments, but with the control characters in column format.
everything	Everything stty knows about is printed. Same as -h option.
speed	The terminal speed alone is reported on the standard output.
size	The terminal (window) sizes are printed on the standard output, first rows and then columns. This option is only appropriate if currently running a window system. size and speed always report on the settings of /dev/tty, and always report the settings to the standard output.

Control Modes

parenb (-parenb)	enable (disable) parity generation and detection.
parext (-parext)	enable (disable) extended parity generation and detection for mark and space parity.
parodd (-parodd)	select odd (even) parity, or mark (space) parity if parext is enabled.
cs5 cs6 cs7 cs8	select character size [see *termio*(7)].

0	hang up line immediately.
110 300 600 1200 1800 2400 4800 9600 19200 exta 38400 extb	Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)
ispeed 0 110 300 600 1200 1800 2400 4800 9600 19200 exta 38400 extb	Set terminal input baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the input baud rate is set to zero, the input baud rate will be specified by the value of the output baud rate.
ospeed 0 110 300 600 1200 1800 2400 4800 9600 19200 exta 38400 extb	Set terminal output baud rate to the number given, if possible. (Not all hardware supports split baud rates.) If the baud rate is set to zero, the line will be hung up immediately.
hupcl (-hupcl)	hang up (do not hang up) connection on last close.
hup (-hup)	same as hupcl (-hupcl).
cstopb (-cstopb)	use two (one) stop bits per character.
cread (-cread)	enable (disable) the receiver.
clocal (-clocal)	assume a line without (with) modem control.
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.
iuclc (-iuclc)	map (do not map) upper-case alphabetics to lower case on input.
ixon (-ixon)	enable (disable) START/STOP output control. Output is stopped by sending an STOP and started by sending an START.
ixany (-ixany)	allow any character (only START) to restart output.

`decctlq (-decctlq)` Same as `-ixany`.

`ixoff (-ixoff)` request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.

`tandem (-tandem)` Same as `ixoff`.

`imaxbel (-imaxbel)` echo (do not echo) BEL when the input line is too long.

`iexten (-iexten)` enable (disable) extended (implementation-defined) functions for input data.

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 alphabetics to upper case on output.

`onlcr (-onlcr)` map (do not map) NL to CR-NL on output.

`ocrnl (-ocrnl)` map (do not map) CR to NL on output.

`onocr (-onocr)` do not (do) output CRs at column zero.

`onlret (-onlret)` on the terminal NL performs (does not perform) the CR function.

`ofill (-ofill)` use fill characters (use timing) for delays.

`ofdel (-ofdel)` fill characters are DELs (NULs).

`cr0 cr1 cr2 cr3` select style of delay for carriage returns [see *termio*(7)].

`nl0 nl1` select style of delay for line-feeds [see *termio*(7)].

`tab0 tab1 tab2 tab3` select style of delay for horizontal tabs [see *termio*(7)].

`bs0 bs1` select style of delay for backspaces [see *termio*(7)].

`ff0 ff1` select style of delay for form-feeds [see *termio*(7)].

`vt0 vt1` select style of delay for vertical tabs [see *termio*(7)].

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).

`cbreak (-cbreak)` Same as `-icanon`.

`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 *not* keep track of column position and, as a result, may be confusing on escaped characters, tabs, and backspaces.

`crterase (-crterase)`
Same as `echoe`.

`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.)

`tostop (-tostop)` send (do not send) SIGTTOU for background processes.

`echoctl (-echoctl)`
echo (do not echo) control characters as ^*char*, delete as ^?

`ctlecho (-ctlecho)`
Same as `echoctl`.

`echoprt (-echoprt)`
echo (do not echo) erase character as character is "erased".

`prterase (-prterase)`
Same as `echoprt`.

`echoke (-echoke)` BS-SP-BS erase (do not BS-SP-BS erase) entire line on line kill.

`crtkill (-crtkill)` Same as `echoke`.

`flusho (-flusho)` output is (is not) being flushed.

`pendin (-pendin)` retype (do not retype) pending input at next read or input character.

`stflush (-stflush)` enable (disable) flush on a synchronous line after every *write*(2). (Does not apply to the 3B2.)

`stappl (-stappl)` use application mode (use line mode) on a synchronous line. (Does not apply to the 3B2.)

Hardware Flow Control Modes

`rtsxoff (-rtsxoff)` enable (disable) RTS hardware flow control on input.

`ctsxon (-ctsxon)` enable (disable) CTS hardware flow control on output.

`dterxoff (-dterxoff)`
enable (disable) DTER hardware flow control on input.

`rlsdxon (-rlsdxon)` enable (disable) RLSD hardware flow control on output.

`isxoff (-isxoff)` enable (disable) isochronous hardware flow control on input.

Clock Modes

`xcibrg` get transmit clock from internal baud rate generator.

`xctset` get the transmit clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.

`xcrset` get transmit clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.

`rcibrg` get receive clock from internal baud rate generator.

`rctset` get receive clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.

`rcrset` get receive clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.

`tsetcoff` transmitter signal element timing clock not provided.

`tsetcrc` output receive clock on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24, clock source.

`tsetcxc` output transmit clock on transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24, clock source.

`rsetcoff` receiver signal element timing clock not provided.

`rsetcrc` output receive clock on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin, clock source.

`rsetcxc` output transmit clock on receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin, clock source.

Control Assignments

control-character c set *control-character* to *c*, where *control-character* is `intr`, `quit`, `erase`, `kill`, `eof`, `eol`, `eol2`, `swtch`, `start`, `stop`, `susp`, `dsusp`, `rprnt`, `flush`, `werase`, `lnext min`, `ctab`, `time`, or `brk`) [`ctab` is used with `-stappl`; `min` and `time` are used with `-icanon`; see *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.
-evenp, or -parity	disable parenb, and set cs8.
even (-even)	Same as evenp (-evenp).
oddp	enable parenb, cs7, and parodd.
-oddp	disable parenb and parodd, and set cs8.
odd (-odd)	Same as oddp (-oddp).
spacep	enable parenb, cs7, and parext.
-spacep	disable parenb and parext, and set cs8.
markp	enable parenb, cs7, parodd, and parext.
-markp	disable parenb, parodd, and parext, and set cs8.
raw (-raw or cooked)	enable (disable) raw input and output (no ERASE, KILL, INTR, 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.
async	set normal asynchronous communications where clock settings are xcibrg, rcibrg, tsetcoff and rsetcoff.
litout (-litout)	Disable (enable) parenb, istrip, and opost, and set cs8 (cs7).
pass8 (-pass8)	Disable (enable) parenb and istrip, and set cs8 (cs7).
crt	Set options for a CRT (echoe, echoctl, and, if >= 1200 baud, echoke.)
dec	Set all modes suitable for Digital Equipment Corp. operating systems users (ERASE, KILL, and INTR characters to ^?, ^U, and ^C, decctlq, and crt.)

Window Size

`rows` *n*	set window size to *n* rows.
`columns` *n*	set window size to *n* columns.
`cols` *n*	An alias for `columns` *n*.
`ypixels` *n*	set vertical window size to *n* pixels.
`xpixels` *n*	set horizontal window size to *n* pixels.

SEE ALSO

`tabs`(1).

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

`termio`(7), `termiox`(7) in the *System Administrator's Reference Manual.*

NAME

sum – calculate a checksum for a file

SYNOPSIS

`/usr/ucb/sum` *filename*

DESCRIPTION

`sum` calculates and displays a 16-bit checksum for the named file, and also displays the size of the file in kilobytes. It is typically used to look for bad spots, or to validate a file communicated over some transmission line. The checksum is calculated by an algorithm which may yield different results on machines with 16-bit `int`s and machines with 32-bit `int`s, so it cannot always be used to validate that a file has been transferred between machines with different-sized `int`s.

SEE ALSO

`wc`(1), `sum`(1) in the *User's Reference Manual.*

DIAGNOSTICS

`Read error` is indistinguishable from EOF on most devices; check the block count.

NOTES

Obsolescent.

NAME

tbl – format tables for nroff or troff

SYNOPSIS

/usr/ucb/tbl [-me] [-ms] [-mm] [-TX] [*filename*] ...

DESCRIPTION

The tbl command is a preprocessor for formatting tables for nroff or troff. The input *filename*s are copied to the standard output, except that lines between .TS and .TE command lines are assumed to describe tables and are reformatted.

If no arguments are given, tbl reads the standard input, so tbl may be used as a filter. When tbl is used with eqn or neqn the tbl command should be first, to minimize the volume of data passed through pipes.

The -me option copies the -me macro package to the front of the output file.
The -ms option copies the -ms macro package to the front of the output file.
The -mm option copies the -mm macro package to the front of the output file.
The -TX option produces output that does not have fractional line motions in it.

EXAMPLE

As an example, letting \t represent a TAB (which should be typed as a genuine TAB) the input

```
.TS
c s s
c c s
c c c
l n n.
Household Population
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.30
Bound Brook\t3425\t3.04
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE
```

yields

Household Population		
Town	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

SEE ALSO

eqn(1), nroff(1), troff(1).

NAME

tcopy – copy a magnetic tape

SYNOPSIS

/usr/ucb/tcopy *source* [*destination*]

DESCRIPTION

tcopy copies the magnetic tape mounted on the tape drive specified by the *source* argument. The only assumption made about the contents of a tape is that there are two tape marks at the end.

When only a source drive is specified, tcopy scans the tape, and displays information about the sizes of records and tape files. If a destination is specified, tcopy makes a copies the source tape onto the *destination* tape, with blocking preserved. As it copies, tcopy produces the same output as it does when only scanning a tape.

SEE ALSO

mt(1)

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

NOTES

tcopy will only run on systems supporting an associated set of ioctl(2) requests.

NAME

`test` – condition evaluation command

SYNOPSIS

`/usr/ucb/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. Alternatively, if `/usr/sh` users specify `/usr/ucb` before `/usr/bin` in their PATH environment variable, then *test* will return true if *file* exists and is (`not-a-directory`). This is also the default for `/usr/bin/csh` users.
`-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` [*fildes*]	true if the open file whose file descriptor number is *fildes* (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**.

-L*file* true if **file** exists and is a symbolic link. With all other primitives, the symbolic links are followed by default.

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) in the *User's Reference Manual.*

NOTES

The 'not-a-directory' alternative to the **-f** option is a transition aid for BSD applications and may not be supported in future releases.

The **-L** option is a migration aid for users of other shells which have similar options and may not be supported in future releases.

If you test a file you own (the *-r*, *-w*, or *-x* tests), but the permission tested does not have the *owner* bit set, a non-zero (false) exit status will be returned even though the file may have the *group* or *other* bit set for that permission. The correct exit status will be set if you are super-user.

The = and **!=** operators have a higher precedence than the **-r** through **-n** operators, and = and **!=** always expect arguments; therefore, = and **!=** cannot be used with the **-r** through **-n** operators.

If more than one argument follows the **-r** through **-n** operators, only the first argument is examined; the others are ignored, unless a **-a** or a **-o** is the second argument.

NAME

`tr` – translate characters

SYNOPSIS

`/usr/ucb/tr [ -cds ] [` *string1* `[` *string2* `] ]`

DESCRIPTION

`tr` copies the standard input to the standard output with substitution or deletion of selected characters. The arguments *string1* and *string2* are considered sets of characters. Any input character found in *string1* is mapped into the character in the corresponding position within *string2*. When *string2* is short, it is padded to the length of *string1* by duplicating its last character.

In either string the notation:

a–b

denotes a range of characters from *a* to *b* in increasing ASCII order. The character \, followed by 1, 2 or 3 octal digits stands for the character whose ASCII code is given by those digits. As with the shell, the escape character \, followed by any other character, escapes any special meaning for that character.

OPTIONS

Any combination of the options `-c`, `-d`, or `-s` may be used:

`-c` Complement the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 01 through 0377 octal.

`-d` Delete all input characters in *string1*.

`-s` Squeeze all strings of repeated output characters that are in *string2* to single characters.

EXAMPLE

The following example creates a list of all the words in *filename1* one per line in *filename2*, where a word is taken to be a maximal string of alphabetics. The second string is quoted to protect '\' from the shell. 012 is the ASCII code for NEWLINE.

```
tr -cs A-Za-z '\012' <filename1>filename2
```

SEE ALSO

`ed`(1) in the *User's Reference Manual.*
`ascii`(5) in the *System Administrator's Reference Manual.*

NOTES

Will not handle ASCII NUL in *string1* or *string2*. `tr` always deletes NUL from input.

NAME

troff – typeset or format documents

SYNOPSIS

/usr/ucb/troff [-afiz] [-F*dir*] [-m*name*] [-n*N*] [-o*list*] [-r*aN*]
[-s*N*] [-T*dest*] [-u*N*] [*filename*] ...

DESCRIPTION

troff formats text in the *filenames*. Input to troff is expected to consist of text interspersed with formatting requests and macros. If no *filename* argument is present, troff reads standard input. A – as a *filename* argument indicates that standard input is to be read at that point in the list of input files; troff reads the files named ahead of the – in the arguments list, then text from the standard input, and then text from the files named after the –.

The following options may appear in any order, but they all must appear before the first *filename*.

-a Send a printable approximation of the formatted output to the standard output file.

-f Do not print a trailer after the final page of output or cause the postprocessor to relinquish control of the device.

-i Read the standard input after the input files are exhausted.

-z Suppress formatted output. Only diagnostic messages and messages output using the .tm request are output.

-F*dir* Search the directory *dir* for font width tables instead of the system-dependent default directory.

-m*name*
Prepend the macro file /usr/lib/tmac/tmac.*name* to the input *filenames*. Note: most references to macro packages include the leading m as part of the name; for example, the man macro package resides in /usr/lib/tmac/tmac.an.

-n*N* Number first generated page *N*.

-o*list* Print only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N–M* means pages *N* through *M*; an initial *–N* means from the beginning to page *N*; and a final *N–* means from *N* to the end.

-r*aN* Set register *a* (one-character) to *N*.

-s*N* Stop the phototypesetter every *N* pages. On some devices, troff produces a trailer so you can change cassettes; resume by pressing the typesetter's start button.

-T*dest* Prepare output for typesetter *dest*. The following values can be supplied for *dest*:

202	Mergenthaler Linotron 202. This is the default value.
cat	Graphics Systems C/A/T.
aps	Autologic APS-5.

-u*N* Set the emboldening factor for the font mounted in position 3 to *N*. If *N* is missing, then set the emboldening factor to 0.

FILES

`/tmp/trtmp`	temporary file
`/usr/ucblib/doctools/tmac/tmac.*`	standard macro files
`/usr/ucblib/doctools/font/*`	font width tables for alternate mounted `troff` fonts

SEE ALSO

checknr(1), chmod(1), eqn(1), lpd(1M), lpr(1), nroff(1), tbl(1), man(7), me(7), ms(7).

chmod(1), col(1) in the *User's Reference Manual.*

NAME

tset, reset – establish or restore terminal characteristics

SYNOPSIS

tset [-InQrs] [-ec] [-kc] [-m [*port*-ID [*baudrate*] : *type*] ...] [*type*]

reset [-] [-ec] [-I] [-kc] [-n] [-Q] [-r] [-s]
[-m [*indent*] [*test baudrate*]: *type*] ... [*type*]

DESCRIPTION

tset sets up your terminal, typically when you first log in. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialized the terminal, and the like. tset first determines the *type* of terminal involved, and then does necessary initializations and mode settings. If a port is not wired permanently to a specific terminal (not hardwired) it is given an appropriate generic identifier such as dialup.

reset clears the terminal settings by turning off CBREAK and RAW modes, output delays and parity checking, turns on NEWLINE translation, echo and TAB expansion, and restores undefined special characters to their default state. It then sets the modes as usual, based on the terminal type (which will probably override some of the above). See stty(1) for more information. All arguments to tset may be used with reset. reset also uses rs= and rf= to reset the initialization string and file. This is useful after a program dies and leaves the terminal in a funny state. Often in this situation, characters will not echo as you type them. You may have to type '<LINEFEED>reset<LINEFEED>' since '<RETURN>' may not work.

When no arguments are specified, tset reads the terminal type from the TERM environment variable and re-initializes the terminal, and performs initialization of mode, environment and other options at login time to determine the terminal type and set up terminal modes.

When used in a startup script (.profile for sh(1) users or .login for csh(1) users) it is desirable to give information about the type of terminal you will usually use on ports that are not hardwired. Any of the alternate generic names given in /etc/termcap may be used for the identifier. Refer to the -m option below for more information. If no mapping applies and a final *type* option, not preceded by a -m, is given on the command line then that type is used.

It is usually desirable to return the terminal type, as finally determined by tset, and information about the terminal's capabilities, to a shell's environment. This can be done using the -, -s, or -S options.

For the Bourne shell, put this command in your .profile file:

```
eval `tset -s options...`
```

or using the C shell, put this command in your .login file:

```
eval `tset -s options...`
```

With the C shell, it is also convenient to make an alias in your `.cshrc` file:

```
alias tset 'eval `tset -s \!*`'
```

This also allows the command:

```
tset 2621
```

to be invoked at any time to set the terminal and environment. It is not possible to get this aliasing effect with a Bourne shell script, because shell scripts cannot set the environment of their parent. If a process could set its parent's environment, none of this nonsense would be necessary in the first place.

Once the terminal type is known, `tset` sets the terminal driver mode. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the line-kill (full line erase)) characters, and setting special character delays. TAB and NEWLINE expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is '#', the erase character is changed as if `-e` had been used.

The following options are available with `tset`:

- The name of the terminal finally decided upon is output on the standard output. This is intended to be captured by the shell and placed in the TERM environment variable.

`-ec` Set the erase character to be the named character *c* on all terminals. Default is the BACKSPACE key on the keyboard, usually `^H` (CTRL-H). The character *c* can either be typed directly, or entered using the circumflex-character notation used here.

`-ic` Set the interrupt character to be the named character *c* on all terminals. Default is `^C` (CTRL-C). The character *c* can either be typed directly, or entered using the circumflex-character notation used here.

`-I` Suppress transmitting terminal-initialization strings.

`-kc` Set the line kill character to be the named character *c* on all terminals. Default is `^U` (CTRL-U). The kill character is left alone if `-k` is not specified. Control characters can be specified by prefixing the alphabetical character with a circumflex (as in CTRL-U) instead of entering the actual control key itself. This allows you to specify control keys that are currently assigned.

`-n` Specify that the new tty driver modes should be initialized for this terminal. Probably useless since `stty new` is the default.

`-Q` Suppress printing the 'Erase set to' and 'Kill set to' messages.

`-r` In addition to other actions, reports the terminal type.

`-s` Output commands to set and export TERM. This can be used with

```
set noglob
eval `tset -s ...`
unset noglob
```

to bring the terminal information into the environment. Doing so makes programs such as vi(1) start up faster. If the SHELL environment variable ends with csh, C shell commands are output, otherwise Bourne shell commands are output.

-m [*port*-ID[*baudrate*]:*type*] ...

Specify (map) a terminal type when connected to a generic port (such as *dialup* or *plugboard*) identified by *port-ID*. The *baudrate* argument can be used to check the baudrate of the port and set the terminal type accordingly. The target rate is prefixed by any combination of the following operators to specify the conditions under which the mapping is made:

>	Greater than
@	Equals or "at"
<	Less than
!	It is not the case that (negates the above operators)
?	Prompt for the terminal type. If no response is given, then *type* is selected by default.

In the following example, the terminal type is set to adm3a if the port is a dialup with a speed of greater than 300 or to dw2 if the port is a dialup at 300 baud or less. In the third case, the question mark preceding the terminal type indicates that the user is to verify the type desired. A NULL response indicates that the named type is correct. Otherwise, the user's response is taken to be the type desired.

```
tset -m 'dialup>300:adm3a' -m 'dialup:dw2' -m \
     'plugboard:?adm3a'
```

To prevent interpretation as metacharacters, the entire argument to -m should be enclosed in single quotes. When using the C shell, exclamation points should be preceded by a backslash (\).

EXAMPLES

These examples all use the '-' option. A typical use of tset in a .profile or .login will also use the -e and -k options, and often the -n or -Q options as well. These options have been omitted here to keep the examples short.

To select a 2621, you might put the following sequence of commands in your .login file (or .profile for Bourne shell users).

```
set noglob
eval `tset -s 2621`
unset noglob
```

If you have a switch which connects to various ports (making it impractical to identify which port you may be connected to), and use various terminals from time to time, you can select from among those terminals according to the *speed* or baud rate. In the example below, tset will prompt you for a terminal type if the baud rate is greater than 1200 (say, 9600 for a terminal connected by an RS-232 line), and use a Wyse® 50 by default. If the baud rate is less than or equal to 1200, it will select a 2621. Note the placement of the question mark, and the quotes to protect the > and ? from interpretation by the shell.

```
set noglob
eval `tset -s -m 'switch>1200:?wy' -m 'switch<=1200:2621'`
unset noglob
```

The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals, and the terminal you use most often is an adm3a.

```
set noglob
eval `tset -s ?adm3a`
unset noglob
```

If you want to make the selection based only on the baud rate, you might use the following:

```
set noglob
eval `tset -s -m '>1200:wy' 2621`
unset noglob
```

The following example quietly sets the erase character to BACKSPACE, and kill to CTRL-U. If the port is switched, it selects a Concept™ 100 for speeds less than or equal to 1200, and asks for the terminal type otherwise (the default in this case is a Wyse 50). If the port is a direct dialup, it selects Concept 100 as the terminal type. If logging in over the ARPANET, the terminal type selected is a Datamedia® 2500 terminal or emulator. Note the backslash escaping the NEWLINE at the end of the first line in the example.

```
set noglob
eval `tset -e -k^U -Q -s -m 'switch<=1200:concept100' -m \
     'switch:?wy' -m dialup:concept100 -m arpanet:dm2500`
unset noglob
```

FILES

```
.login
.profile
```

SEE ALSO

csh(1), sh(1), vi(1), stty(1) in the *User's Reference Manual.*

ttytab(5), termcap(5), environ(5) in the *System Administrator's Reference Manual.*

NOTES

The tset command is one of the first commands a user must master when getting started on a UNIX system. Unfortunately, it is one of the most complex, largely because of the extra effort the user must go through to get the environment of the login shell set. Something needs to be done to make all this simpler, either the login program should do this stuff, or a default shell alias should be made, or a way to set the environment of the parent should exist.

This program cannot intuit personal choices for erase, interrupt and line kill characters, so it leaves these set to the local system standards.

It could well be argued that the shell should be responsible for ensuring that the terminal remains in a sane state; this would eliminate the need for the reset program.

NAME

ul – underline

SYNOPSIS

/usr/ucb/ul [-i] [-t *terminal*] [*filename*...]

DESCRIPTION

The ul command reads the named *filenames* (or the standard input if none are given) and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable TERM. ul uses the /usr/share/lib/termcap file to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode then that is used instead. If the terminal can overstrike, or handles underlining automatically, ul degenerates to cat. If the terminal cannot underline, underlining is ignored.

The following options are available:

-t *terminal* Override the terminal kind specified in the environment. If the terminal cannot underline, underlining is ignored.

-i Indicate underlining by a separate line containing appropriate dashes –; this is useful when you want to look at the underlining which is present in an nroff output stream on a CRT-terminal.

SEE ALSO

man(1), nroff(1)

cat(1) in the *User's Reference Manual.*

NAME

unifdef – resolve and remove ifdef'ed lines from C program source

SYNOPSIS

/usr/ucb/unifdef [-clt] [-D*name*] [-U*name*] [-iD*name*] [-iU*name*] ... [*filename*]

DESCRIPTION

unifdef removes ifdefed lines from a file while otherwise leaving the file alone. It is smart enough to deal with the nested ifdefs, comments, single and double quotes of C syntax, but it does not do any including or interpretation of macros. Neither does it strip out comments, though it recognizes and ignores them. You specify which symbols you want defined with -D options, and which you want undefined with -U options. Lines within those ifdefs will be copied to the output, or removed, as appropriate. Any ifdef, ifndef, else, and endif lines associated with *filename* will also be removed.

ifdefs involving symbols you do not specify are untouched and copied out along with their associated ifdef, else, and endif lines.

If an ifdef*X* occurs nested inside another ifdef*X*, then the inside ifdef is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence is significant.

unifdef copies its output to the standard output and will take its input from the standard input if no *filename* argument is given.

The following options are available:

-c Complement the normal operation. Lines that would have been removed or blanked are retained, and vice versa.

-l Replace "lines removed" lines with blank lines.

-t Plain text option. unifdef refrains from attempting to recognize comments and single and double quotes.

-iD*name* Ignore, but print out, lines associated with the defined symbol *name*. If you use ifdefs to delimit non-C lines, such as comments or code which is under construction, then you must tell unifdef which symbols are used for that purpose so that it will not try to parse for quotes and comments within them.

-iU*name* Ignore, but print out, lines associated with the undefined symbol *name*.

SEE ALSO

cc(1)

cc(1) in the *Programmer's Reference Manual.*
diff(1) in the *User's Reference Manual.*

DIAGNOSTICS

Premature EOF Inappropriate else or endif.

Exit status is 0 if output is exact copy of input, 1 if not, 2 if unifdef encounters problems.

NAME

uptime – show how long the system has been up

SYNOPSIS

/usr/ucb/uptime

DESCRIPTION

The uptime command prints the current time, the length of time the system has been up. It is the first line of a w(1) command.

EXAMPLE

Below is an example of the output uptime provides:

```
uptime
6:47am  up 6 days, 16:38,  1 users
```

SEE ALSO

w(1)

whodo(1) in the *System Administrator's Reference Manual.*
who in the *User's Reference Manual.*

NOTES

who –b gives the time the system was last booted.

NAME

users – display a compact list of users logged in

SYNOPSIS

/usr/ucb/users [*file*]

DESCRIPTION

users lists the login names of the users currently on the system in a compact, one-line format.

Specifying *file*, tells users where to find its information; by default it checks /var/adm/utmp.

Typing *users* is equivalent to typing who -q.

EXAMPLE

```
users
paul george ringo
```

FILES

/var/adm/utmp

SEE ALSO

who(1) in the *User's Reference Manual.*

NAME

vacation – reply to mail automatically

SYNOPSIS

/usr/ucb/vacation [-I]
/usr/ucb/vacation [-j] [-a*alias*] [-t*N*] *username*

DESCRIPTION

vacation automatically replies to incoming mail. The reply is contained in the file .vacation.msg, that you create in your home directory.

This file should include a header with at least a 'Subject:' line (it should not include a 'From:' or a 'To:' line). For example:

```
Subject: I am on vacation
I am on vacation until July 22.  If you have something urgent,
please contact Joe Jones (jones@f40).
        --John
```

If the string $SUBJECT appears in the .vacation.msg file, it is replaced with the subject of the original message when the reply is sent; thus, a .vacation.msg file such as

```
Subject: I am on vacation
I am on vacation until July 22.
Your mail regarding "$SUBJECT" will be read when I return.
If you have something urgent, please contact
Joe Jones (jones@f40).
        --John
```

will include the subject of the message in the reply.

No message is sent if the 'To:' or the 'Cc:' line does not list the user to whom the original message was sent or one of a number of aliases for them, if the initial From line includes the string -REQUEST@, or if a 'Precedence: bulk' or 'Precedence: junk' line is included in the header.

The following options are available:

-I Initialize the .vacation.pag and .vacation.dir files and start /usr/ucb/vacation.

If the -I flag is not specified, and a *user* argument is given, /usr/ucb/vacation reads the first line from the standard input (for a 'From:' line, no colon). If absent, it produces an error message. The following options may be specified:

-a*alias*
Indicate that *alias* is one of the valid aliases for the user running /usr/ucb/vacation, so that mail addressed to that alias generates a reply.

-j Do not check whether the recipient appears in the 'To: ' or the 'Cc:' line.

-t*N* Change the interval between repeat replies to the same sender. The default is 1 week. A trailing `s`, `m`, `h`, `d`, or `w` scales *N* to seconds, minutes, hours, days, or weeks respectively.

USAGE

To start `/usr/ucb/vacation`, create a `.forward` file in your home directory containing a line of the form:

username, "|`/usr/ucb/vacation` *username*"

where *username* is your login name.

Then type in the command:

```
/usr/ucb/vacation -I
```

To stop `/usr/ucb/vacation`, remove the `.forward` file, or move it to a new name.

If `/usr/ucb/vacation` is run with no arguments, it will permit you to interactively turn `/usr/ucb/vacation` on or off. It will create a `.vacation.msg` file for you, or edit an existing one, using the editor specified by the `VISUAL` or `EDITOR` environment variable, or `vi`(1) if neither of those environment variables are set. If a `.forward` file is present in your home directory, it will ask whether you want to remove it and turn off `/usr/ucb/vacation`. If it is not present in your home directory, it creates it for you, and automatically performs a '`/usr/ucb/vacation -I`' function, turning on `/usr/ucb/vacation`.

FILES

```
~/.forward
~/.vacation.mesg
```

A list of senders is kept in the files `.vacation.pag` and `.vacation.dir` in your home directory.

SEE ALSO

`sendmail`(1M).

`vi`(1) in the *User's Reference Manual.*

NAME

w – who is logged in, and what are they doing

SYNOPSIS

/usr/ucb/w [-hls] [*user*]

DESCRIPTION

The w command displays a summary of the current activity on the system, including what each user is doing. The heading line shows the current time of day, how long the system has been up, and the number of users logged into the system.

The fields displayed are: the users login name, the name of the tty the user is on, the time of day the user logged on (in *hours:minutes*), the idle time—that is, the number of minutes since the user last typed anything (in *hours:minutes*), the CPU time used by all processes and their children on that terminal (in *minutes:seconds*), the CPU time used by the currently active processes (in *minutes:seconds*), the name and arguments of the current process.

If a *user* name is included, output is restricted to that user.

The following options are available:

-h Suppress the heading.

-l Produce a long form of output, which is the default.

-s Produce a short form of output. In the short form, the tty is abbreviated, the login time and CPU times are left off, as are the arguments to commands.

EXAMPLE

```
w
7:36am  up 6 days, 16:45,  1 users
User tty  login@     idle JCPU PCPU what
ralph console    7:10am        1 10:05 4:31 w
```

FILES

/var/adm/utmp
/dev/kmem
/dev/drum

SEE ALSO

ps(1), who(1) in the *User's Reference Manual.*
utmp(4), whodo(1M) in the *System Administrator's Reference Manual.*

NOTES

The notion of the "current process" is muddy. The current algorithm is 'the highest numbered process on the terminal that is not ignoring interrupts, or, if there is none, the highest numbered process on the terminal'. This fails, for example, in critical sections of programs like the shell and editor, or when faulty programs running in the background fork and fail to ignore interrupts. In cases where no process can be found, w prints –.

The CPU time is only an estimate, in particular, if someone leaves a background process running after logging out, the person currently on that terminal is "charged" with the time.

Background processes are not shown, even though they account for much of the load on the system.

Sometimes processes, typically those in the background, are printed with null or garbaged arguments. In these cases, the name of the command is printed in parentheses.

w does not know about the conventions for detecting background jobs. It will sometimes find a background job instead of the right one.

NAME

whatis – display a one-line summary about a keyword

SYNOPSIS

/usr/ucb/whatis *command...*

DESCRIPTION

whatis looks up a given *command* and displays the header line from the manual section. You can then run the man(1) command to get more information. If the line starts *'name(section) ...'* you can do *'man section name'* to get the documentation for it. Try 'whatis ed' and then you should do 'man 1 ed' to get the manual page for ed(1).

whatis is actually just the -f option to the man command.

FILES

/usr/share/man/whatis data base

SEE ALSO

man(1), catman(1M)

NAME

which – locate a command; display its pathname or alias

SYNOPSIS

/usr/ucb/which [*filename*] ...

DESCRIPTION

which takes a list of names and looks for the files which would be executed had these names been given as commands. Each argument is expanded if it is aliased, and searched for along the user's path. Both aliases and path are taken from the user's .cshrc file.

FILES

~/.cshrc	source of aliases and path values

SEE ALSO

csh(1), ksh(1), sh(1) in the *User's Reference Manual.*

DIAGNOSTICS

A diagnostic is given for names which are aliased to more than a single word, or if an executable file with the argument name was not found in the path.

NOTES

Only aliases and paths from ~/.cshrc are used; importing from the current environment is not attempted.

which must be executed by csh(1), since only csh knows about aliases. If you are using sh instead of csh, whence -v provides similar functionality.

To compensate for ~/.cshrc files in which aliases depend upon the prompt variable being set, which sets this variable. If the ~/.cshrc produces output or prompts for input when prompt is set, which may produce some strange results.

NAME

whoami – display the effective current username

SYNOPSIS

/usr/ucb/whoami

DESCRIPTION

whoami displays the login name corresponding to the current effective user ID. If you have used su to temporarily adopt another user, whoami will report the login name associated with that user ID. whoami gets its information from the geteuid and getpwuid library routines (see getuid and getpwent, respectively).

FILES

/etc/passwd username data base

SEE ALSO

su(1), who(1) in the *User's Reference Manual.*
getuid(2), getpwent(3) in the *Programmer's Reference Manual.*

NAME

catman – create the cat files for the manual

SYNOPSIS

/usr/ucb/catman [-nptw] [-M *directory*] [-T *tmac.an*] [*sections*]

DESCRIPTION

The catman commands creates the preformatted versions of the on-line manual from the nroff(1) input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, catman recreates the whatis database.

If there is one parameter not starting with a '-', it is taken to be a list of manual sections to look in. For example

```
catman 123
```

only updates manual sections 1, 2, and 3.

The following options are available:

-n Do not (re)create the whatis database.

-p Print what would be done instead of doing it.

-t Create troffed entries in the appropriate fmt subdirectories instead of nroffing into the cat subdirectories.

-w Only create the whatis database. No manual reformatting is done.

-M Update manual pages located in the specified directory (/usr/share/man by default).

-T Use tmac.an in place of the standard manual page macros.

ENVIRONMENT

TROFF The name of the formatter to use when the -t flag is given. If not set, 'troff' is used.

FILES

/usr/share/man	default manual directory location
/usr/share/man/man?/*.*	raw (nroff input) manual sections
/usr/share/man/cat?/*.*	preformatted nroffed manual pages
/usr/share/man/fmt?/*.*	preformatted troffed manual pages
/usr/share/man/whatis	whatis database location
/usr/ucblib/makewhatis	command script to make whatis database

SEE ALSO

man(1), nroff(1), troff(1), whatis(1)

DIAGNOSTICS

man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory
The file outside the parentheses is missing, and is referred to by the file inside them.

target of .so in man?/xxx.? must be relative to /usr/man
catman only allows references to filenames that are relative to the directory /usr/share/man.

`opendir:man?: No such file or directory`
A harmless warning message indicating that one of the directories `catman` normally looks for is missing.

`*.*: No such file or directory`
A harmless warning message indicating `catman` came across an empty directory.

NAME

du – display the number of disk blocks used per directory or file

SYNOPSIS

/usr/ucb/du [-F ufs]

/usr/ucb/du [-F ufs] [-a] [-s] [*filename* . . .]

DESCRIPTION

du gives the number of kilobytes contained in all files and, recursively, directories within each specified directory or file *filename*. If *filename* is missing, '.' (the current directory) is used.

A file which has multiple links to it is only counted once.

OPTIONS

-a Generate an entry for each file.

-s Only display the grand total for each of the specified *filenames*.

Entries are generated only for each directory in the absence of options.

EXAMPLE

Here is an example of using du in a directory. We used the pwd(1) command to identify the directory, then used du to show the usage of all the subdirectories in that directory. The grand total for the directory is the last entry in the display:

```
% pwd
/usr/ralph/misc
% du
5     ./jokes
33    ./squash
44    ./tech.papers/lpr.document
217   ./tech.papers/new.manager
401   ./tech.papers
144   ./memos
80    ./letters
388   ./window
93    ./messages
15    ./useful.news
1211  .
%
```

SEE ALSO

df(1M), pwd(1) in the *User's Reference Manual.*
quot(1M) in the *System Administrator's Reference Manual.*

NOTES

Filename arguments that are not directory names are ignored, unless you use -a.

If there are too many distinct linked files, du will count the excess files more than once.

NAME

fastboot, fasthalt – reboot/halt the system without checking the disks

SYNOPSIS

/usr/ucb/fastboot [*boot-options*]

/usr/ucb/fasthalt [*halt-options*]

DESCRIPTION

fastboot and fasthalt are shell scripts that invoke reboot and halt with the proper arguments.

These commands are provided for compatibility only.

FILES

/etc/rc

SEE ALSO

halt(1M), reboot(1M).

fsck(1M), init(1M), rc0(1M), rc2(1M), rc6(1M) in the *System Administrator's Reference Manual.*

NAME

grpck – check group database entries

SYNOPSIS

/usr/ucb/grpck [*filename*]

DESCRIPTION

grpck checks that a file in group(4) does not contain any errors; it checks the /etc/group file by default.

This command differs from /usr/sbin/grpck in its ability to correctly parse YP entries in /etc/passwd.

FILES

/etc/group

SEE ALSO

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

DIAGNOSTICS

`Too many/few fields`
: An entry in the group file does not have the proper number of fields.

`No group name`
: The group name field of an entry is empty.

`Bad character(s) in group name`
: The group name in an entry contains characters other than lower-case letters and digits.

`Invalid GID`
: The group ID field in an entry is not numeric or is greater than 65535.

`Null login name`
: A login name in the list of login names in an entry is null.

`Login name not found in password file`
: A login name in the list of login names in an entry is not in the password file.

NAME

halt – stop the processor

SYNOPSIS

/usr/ucb/halt [-nqy]

DESCRIPTION

halt writes out any information pending to the disks and then stops the processor.

halt normally logs the system shutdown to the system log daemon, syslogd(1M), and places a shutdown record in the login accounting file /var/adm/wtmp. These actions are inhibited if the -n or -q options are present.

The following options are available:

-n Prevent the *sync* before stopping.

-q Quick halt. No graceful shutdown is attempted.

-y Halt the system, even from a dialup terminal.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

reboot(1M), syslogd(1M)

shutdown(1M), init(1M) in the *System Administrator's Reference Manual.*

NOTES

This command is equivalent to init 0.

NAME

lpc – line printer control program

SYNOPSIS

/usr/ucb/lpc [*command* [*parameter*...]]

DESCRIPTION

lpc controls the operation of the printer, or of multiple printers. lpc commands can be used to start or stop a printer, disable or enable a printer's spooling queue, rearrange the order of jobs in a queue, or display the status of each printer—along with its spooling queue and printer daemon.

With no arguments, lpc runs interactively, prompting with 'lpc>'. If arguments are supplied, lpc interprets the first as a *command* to execute; each subsequent argument is taken as a *parameter* for that command. The standard input can be redirected so that lpc reads commands from a file.

Commands may be abbreviated to an unambiguous substring. Note: the *printer* parameter is specified just by the name of the printer (as lw), not as you would specify it to lpr(1) or lpq(1) (not as -Plw).

? [*command*]...

help [*command*]...

Display a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

abort [all | [*printer*...]]

Terminate an active spooling daemon on the local host immediately and then disable printing (preventing new daemons from being started by lpr(1)) for the specified printers. The abort command can only be used by the privileged user.

clean [all | [*printer*...]]

Remove all files created in the spool directory by the daemon from the specified printer queue(s) on the local machine. The clean command can only be used by the privileged user.

disable [all | [*printer*...]]

Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by lpr(1). The disable command can only be used by the privileged user.

down [all | [*printer*...]] [*message*]

Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message does not need to be quoted, the remaining arguments are treated like echo(1). This is normally used to take a printer down and let others know why (lpq(1) indicates that the printer is down, as does the status command).

enable [all | [*printer*...]]

Enable spooling on the local queue for the listed printers, so that lpr(1) can put new jobs in the spool queue. The enable command can only be used by the privileged user.

`exit`
`quit` Exit from `lpc`.

`restart [all | [printer...]]`
Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. This command can be run by any user.

`start [all | [printer...]]`
Enable printing and start a spooling daemon for the listed printers. The `start` command can only be used by the privileged user.

`status [all | [printer...]]`
Display the status of daemons and queues on the local machine. This command can be run by any user.

`stop [all | [printer...]]`
Stop a spooling daemon after the current job completes and disable printing. The `stop` command can only be used by the privileged user.

`topq` *printer* [*job#*...] [*user*...]
Move the print job(s) specified by *job#* or those job(s) belonging to *user* to the top (head) of the printer queue. The `topq` command can only be used by the privileged user.

`up [all | [printer...]]` Enable everything and start a new printer daemon. Undoes the effects of `down`.

FILES

`/var/spool/lp/*`
`/var/spool/lp/system/pstatus`

SEE ALSO

`lpq`(1), `lpr`(1), `lprm`(1)

`echo`(1) in the *User's Reference Manual.*
`lpsched`(1M) in the *System Administrator's Reference Manual.*

DIAGNOSTICS

`?Ambiguous command`
The abbreviation you typed matches more than one command.

`?Invalid command`
You typed a command or abbreviation that was not recognized.

`?Privileged command`
You used a command can be executed only by the privileged user.

`lpc: printer: unknown printer to the print service`
The `printer` was not found in the System V LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use '`lptstat -p`' to find the reason.

`lpc: error on opening queue to spooler`
The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon `/usr/lib/lp/lpsched` is running.

`lpc: Can't send message to LP print service`

`lpc: Can't receive message from LP print service`
These indicate that the LP print service has been stopped. Get help from the system administrator.

`lpc: Received unexpected message from LP print service`
It is likely there is an error in this software. Get help from system administrator.

NAME

mailstats – print statistics collected by sendmail

SYNOPSIS

/usr/ucb/mailstats [*filename*]

DESCRIPTION

mailstats prints out the statistics collected by the sendmail program on mailer usage. These statistics are collected if the file indicated by the S configuration option of sendmail exists. mailstats first prints the time that the statistics file was created and the last time it was modified. It will then print a table with one row for each mailer specified in the configuration file. The first column is the mailer number, followed by the symbolic name of the mailer. The next two columns refer to the number of messages received by *sendmail,* and the last two columns refer to messages sent by *sendmail.* The number of messages and their total size (in 1024 byte units) is given. No numbers are printed if no messages were sent (or received) for any mailer.

You might want to add an entry to /var/spool/cron/crontab/root to reinitialize the statistics file once a night. Copy /dev/null into the statistics file or otherwise truncate it to reset the counters.

FILES

/var/spool/cron/crontab/root
/dev/null

SEE ALSO

sendmail(1M).

NOTES

mailstats should read the configuration file instead of having a hard-wired table mapping mailer numbers to names.

NAME

newaliases – rebuild the data base for the mail aliases file

SYNOPSIS

/usr/ucb/newaliases

DESCRIPTION

newaliases rebuilds the random access data base for the mail aliases file /etc/aliases. It is run automatically by sendmail(1M) (in the default configuration) whenever a message is sent.

FILES

/etc/aliases

SEE ALSO

sendmail(1M), aliases(4).

NAME

newfs – construct a new file system

SYNOPSIS

/usr/ucb/newfs [-nNv] [*mkfs-options*] *block-special-file*

DESCRIPTION

newfs is a friendly front-end to the mkfs(1M) program. On Sun systems, the disk type is determined by reading the disk label for the specified *block-special-file*.

block-special-file is the name of a block special device residing in /dev. If you want to make a file system on sd0, you can specify sd0 rsd0 or /dev/rsd0; if you only specify sd0, newfs will find the proper device.

newfs then calculates the appropriate parameters to use in calling mkfs, builds the file system by forking mkfs and, if the file system is a root partition, installs the necessary bootstrap programs in its initial 16 sectors.

OPTIONS

-n Do not install the bootstrap programs.

-N Print out the file system parameters without actually creating the file system.

-v Verbose. newfs prints out its actions, including the parameters passed to mkfs.

mkfs-options

Options that override the default parameters passed to mkfs(1M) are:

-b *block-size*

The block size of the file system in bytes.

-c *#cylinders/group*

The number of cylinders per cylinder group in a file system. The default value used is 16.

-d *rotdelay*

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file.

-f *frag-size*

The fragment size of the file system in bytes.

-i *bytes/inode*

This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

-m *free-space%*

The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

`-o` *optimization*
(`space` or `time`). The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the minimum free space threshold (as specified by the `-m` option) is less than 10%, the default is to optimize for space; if the minimum free space threshold is greater than or equal to 10%, the default is to optimize for time.

`-r` *revolutions/minute*
The speed of the disk in revolutions per minute (normally 3600).

`-s` *size*
The size of the file system in sectors.

`-t` *#tracks/cylinder*
The number of tracks per cylinders on the disk.

FILES

`/usr/mdec` for boot strapping programs
`/dev`

SEE ALSO

`fsck`(1M), `mkfs`(1M), `tunefs`(1M), `fs`(4) in the *System Administrator's Reference Manual.*

NAME

pwck – check password database entries

SYNOPSIS

/usr/ucb/pwck [*filename*]

DESCRIPTION

pwck checks a password file for errors. If specified, *filename* is checked, otherwise /etc/passwd is checked.

This command differs from /usr/sbin/pwck in its ability to correctly parse YP entries in /etc/passwd.

DIAGNOSTICS

`Too many/few fields`
: An entry in the password file does not have the proper number of fields.

`No login name`
: The login name field of an entry is empty.

`Bad character(s) in login name`
: The login name in an entry contains characters other than lower-case letters and digits.

`First char in login name not lower case alpha`
: The login name in an entry does not begin with a lower-case letter.

`Login name too long`
: The login name in an entry has more than 8 characters.

`Invalid UID`
: The user ID field in an entry is not numeric or is greater than 65535.

`Invalid GID`
: The group ID field in an entry is not numeric or is greater than 65535.

`No login directory`
: The login directory field in an entry is empty.

`Login directory not found`
: The login directory field in an entry refers to a directory that does not exist.

`Optional shell file not found.`
: The login shell field in an entry refers to a program or shell script that does not exist.

`No netgroup name`
: The entry is a Yellow Pages entry referring to a netgroup, but no netgroup is present.

`Bad character(s) in netgroup name`
: The netgroup name in a Yellow Pages entry contains characters other than lower-case letters and digits.

`First char in netgroup name not lower case alpha`
: The netgroup name in a Yellow pages entry does not begin with a lower-case letter.

FILES

`/etc/passwd`

SEE ALSO

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

NAME

reboot – restart the operating system

SYNOPSIS

/usr/ucb/reboot [-dnq] [*boot arguments*]

DESCRIPTION

reboot restarts the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to it.

Although reboot can be run by the privileged user at any time, shutdown(1M) is normally used first to warn all users logged in of the impending loss of service. See shutdown(1M) for details.

reboot performs a sync(1) operation on the disks, and then a multiuser reboot is initiated. See init(1M) for details.

reboot normally logs the reboot to the system log daemon, syslogd(1M), and places a shutdown record in the login accounting file /var/adm/wtmp. These actions are inhibited if the -n or -q options are present.

The following options are available:

-d Dump system core before rebooting. This option is provided for compatibility, but is not supported by the underlying reboot(3) call.

-n Avoid the sync(1). It can be used if a disk or the processor is on fire.

-q Quick. Reboots quickly and ungracefully, without first shutting down running processes.

boot arguments

These arguments are accepted for compatibility, but are ignored by reboot. See boot(1M) for details.

Power Fail and Crash Recovery

Normally, the system will reboot itself at power-up or after crashes.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

halt(1M), syslogd(1M), reboot(3).

boot(8), crash(1M), fsck(1M), init(1M), shutdown(1M), sync(1M), in the *System Administrator's Reference Manual.*

NAME

renice – alter priority of running processes

SYNOPSIS

/usr/ucb/renice *priority pid ...*

/usr/ucb/renice *priority* [-p *pid ...*] [-g *pgrp ...*] [-u *username ...*]

DESCRIPTION

The renice command alters the scheduling priority of one or more running processes. By default, the processes to be affected are specified by their process IDs. *priority* is the new priority value.

The following options are available:

-p *pid ...* Specify a list of process IDs.

-g *pgrp ...* Specify a list of process group IDs. The processes in the specified process groups have their scheduling priority altered.

-u *user ...* Specify a list of user IDs or usernames. All processes owned by each *user* have their scheduling altered.

Users other than the privileged user may only alter the priority of processes they own, and can only monotonically increase their nice value within the range 0 to 20. This prevents overriding administrative fiats. The privileged user may alter the priority of any process and set the priority to any value in the range −20 to 20. Useful priorities are: 19 (the affected processes will run only when nothing else in the system wants to), 0 (the base scheduling priority) and any negative value (to make things go very fast).

If only the priority is specified, the current process (alternatively, process group or user) is used.

FILES

/etc/passwd map user names to user ID's

SEE ALSO

priocntl(1) in the *User's Reference Manual.*

NOTES

If you make the priority very negative, then the process cannot be interrupted.

To regain control you must make the priority greater than zero.

Users other than the privileged user cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

The priocntl command subsumes the function of renice.

NAME

sendmail – send mail over the internet

SYNOPSIS

/usr/ucb/sendmail [-ba] [-bd] [-bi] [-bm] [-bp] [-bs] [-bt] [-bv] [-bz] [-C*file*] [-d*X*] [-F*fullname*] [-f*name*] [-h*N*] [-n] [-o x*value*] [-q[*time*]] [-r*name*] [-t] [-v] [*address* ...]

DESCRIPTION

sendmail sends a message to one or more people, routing the message over whatever networks are necessary. sendmail does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; sendmail is used only to deliver pre-formatted messages.

With no flags, sendmail reads its standard input up to an EOF, or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local aliases(4) file, or by using the YP name service, and aliased appropriately. In addition, if there is a .forward file in a recipient's home directory, sendmail forwards a copy of each message to the list of recipients that file contains. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in alias expansions, for example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

sendmail will also route mail directly to other known hosts in a local network. The list of hosts to which mail is directly sent is maintained in the file /usr/lib/mailhosts.

The following options are available:

-ba Go into ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.

-bd Run as a daemon, waiting for incoming SMTP connections.

-bi Initialize the alias database.

-bm Deliver mail in the usual way (default).

-bp Print a summary of the mail queue.

-bs Use the SMTP protocol as described in RFC 821. This flag implies all the operations of the -ba flag that are compatible with SMTP.

-bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

-bv Verify names only — do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

-bz Create the configuration freeze file.

-C*file* Use alternate configuration file.

-d*X* Set debugging value to *X*.

-F*fullname*
Set the full name of the sender.

-f*name*
Sets the name of the "from" person (that is, the sender of the mail). -f can only be used by "trusted" users (who are listed in the config file).

-h*N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

-M*id* Attempt to deliver the queued message with message-id id.

-n Do not do aliasing.

-o*x value*
Set option *x* to the specified *value*. Options are described below.

-q[*time*]
Processed saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *Time* is given as a tagged number, with s being seconds, m being minutes, h being hours, d being days, and w being weeks. For example, -q1h30m or -q90m would both set the timeout to one hour thirty minutes.

-r*name*
An alternate and obsolete form of the -f flag.

-R*string*
Go through the queue of pending mail and attempt to deliver any message with a recipient containing the specified string. This is useful for clearing out mail directed to a machine which has been down for awhile.

-t Read message for recipients. "To:", "Cc:", and "Bcc:" lines will be scanned for people to send to. The "Bcc:" line will be deleted before transmission. Any addresses in the argument list will be suppressed.

-v Go into verbose mode. Alias expansions will be announced, etc.

PROCESSING OPTIONS

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the -o flag or in the configuration file. The options are:

A*file* Use alternate alias file.

c On mailers that are considered "expensive" to connect to, do not initiate immediate connection. This requires queueing.

d*x* Set the delivery mode to *x*. Delivery modes are i for interactive (synchronous) delivery, b for background (asynchronous) delivery, and q for queue only — that is, actual delivery is done the next time the queue is run.

D Run newaliases(1M) to automatically rebuild the alias database, if necessary.

e*x* Set error processing to mode *x*. Valid modes are `m` to mail back the error message, `w` to "write" back the error message (or mail it back if the sender is not logged in), `p` to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes `m` or `w` and if the sender is local to this machine, a copy of the message is appended to the file `dead.letter` in the sender's home directory.

F*mode* The mode to use when creating temporary files.

f Save UNIX-system-style "From" lines at the front of messages.

g*N* The default group ID to use when calling mailers.

H*file* The SMTP help file.

i Do not take dots on a line by themselves as a message terminator.

L*n* The log level.

m Send to "me" (the sender) also if I am in an alias expansion.

o If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Q*queuedir*
Select the directory in which to queue messages.

r*timeout*
The timeout on reads; if none is set, `sendmail` will wait forever for a mailer.

S*file* Save statistics in the named file.

s Always instantiate the queue file, even under circumstances where it is not strictly necessary.

T*time* Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

t*stz,dtz*
Set the name of the time zone.

u*N* Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep `sendmail` from suppressing the blanks from between arguments.

`sendmail` returns an exit status describing what it did. The codes are defined in *<sysexits.h>*

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as cannot fork.
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as *newaliases*, `sendmail` rebuilds the alias database. If invoked as *mailq*, `sendmail` prints the contents of the mail queue.

FILES

Except for `/etc/sendmail.cf`, these pathnames are all specified in `/etc/sendmail.cf`. Thus, these values are only approximations.

`/usr/bin/uux`	to deliver uucp mail
`/usr/bin/mail`	to deliver local mail
`/var/spool/mqueue/*`	temp files and queued mail
`~/.forward`	list of recipients for forwarding messages

SEE ALSO

`biff`(1), `aliases`(4).

Su, Zaw-Sing, and Jon Postel, *The Domain Naming Convention for Internet User Applications*, RFC 819, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Postel, Jon, *Simple Mail Transfer Protocol*, RFC 821, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Crocker, Dave, *Standard for the Format of ARPA-Internet Text Messages*, RFC 822, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

NAME

shutdown – close down the system at a given time

SYNOPSIS

/usr/ucb/shutdown [-fhknr] [*time* [*warning-message* ...]

DESCRIPTION

shutdown provides an automated procedure to notify users when the system is to be shut down. *time* specifies when shutdown will bring the system down; it may be the word now (indicating an immediate shutdown), or it may specify a future time in one of two formats: *+number* and *hour:min*. The first form brings the system down in *number* minutes, and the second brings the system down at the time of day indicated in 24-hour notation.

At intervals that get closer as the apocalypse approaches, warning messages are displayed at terminals of all logged-in users, and of users who have remote mounts on that machine. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating /etc/nologin and writing a message there. If this file exists when a user attempts to log in, login(1M) prints its contents and exits. The file is removed just before shutdown exits.

At shutdown time a message is written to the system log daemon, syslogd(1M), containing the time of shutdown, the instigator of the shutdown, and the reason. Then a terminate signal is sent to init, which brings the system down to single-user mode.

The time of the shutdown and the warning message are placed in /etc/nologin, which should be used to inform the users as to when the system will be back up, and why it is going down (or anything else).

OPTIONS

As an alternative to the above procedure, these options can be specified:

- -f Arrange, in the manner of fastboot(1M), that when the system is rebooted, the file systems will not be checked.
- -h Execute halt(1M).
- -k Simulate shutdown of the system. Do not actually shut down the system.
- -n Prevent the normal sync(2) before stopping.
- -r Execute reboot(1M).

FILES

/etc/nologin	tells login not to let anyone log in
/etc/xtab	list of remote hosts that have mounted this host

SEE ALSO

fastboot(1M), halt(1M), reboot(1M), syslogd(1M).

login(1) in the *User's Reference Manual.*
sync(2) in the *Programmer's Reference Manual.*

NOTES

Only allows you to bring the system down between now and 23:59 if you use the absolute time for shutdown.

NAME

syslogd – log system messages

SYNOPSIS

/usr/ucb/syslogd [-d] [-f*configfile*] [-m *interval*]

DESCRIPTION

syslogd reads and forwards system messages to the appropriate log files and/or users, depending upon the priority of a message and the system facility from which it originates. The configuration file /etc/syslog.conf [see syslog.conf(5)] controls where messages are forwarded. syslogd logs a mark (timestamp) message every *interval* minutes (default 20) at priority LOG_INFO to the facility whose name is given as mark in the syslog.conf file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (<>); priorities are defined in sys/syslog.h.

syslogd reads from the STREAMS log driver, /dev/log, from any transport provider specified in /etc/netconfig, /etc/net/*transport*/hosts, and /etc/net/*transport*/services, and from the special device /dev/klog (for kernel messages).

syslogd reads the configuration file when it starts up, and again whenever it receives a HUP signal, at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. syslogd exits when it receives a TERM signal.

As it starts up, syslogd creates the file /etc/syslog.pid, if possible, containing its process ID (PID).

The following options are available:

-d	Turn on debugging.
-f*configfile*	Specify an alternate configuration file.
-m *interval*	Specify an interval, in minutes, between mark messages.

FILES

/etc/syslog.conf	configuration file
/etc/syslog.pid	process ID
/dev/log	STREAMS log driver
/etc/netconfig	specifies the transport providers available on the system
/etc/net/*transport*/hosts	network hosts for each transport
/etc/net/*transport*/services	network services for each transport

SEE ALSO

logger(1), syslog(3), syslog.conf(5)

log(7) in the *System Administrator's Reference Manual.*

NAME

alloca - memory allocator

SYNOPSIS

```
cc [ flag... ] file ... -lucb
#include <alloca.h>
char *alloca(size)
int size;
```

DESCRIPTION

alloca allocates *size* bytes of space in the stack frame of the caller, and returns a pointer to the allocated block. This temporary space is automatically freed when the caller returns. Note: if the allocated block is beyond the current stack limit, the resulting behavior is undefined.

SEE ALSO

sigstack(3), sigvec(3).

csh(1) in the *User's Reference Manual.*
ld(1), brk(2), getrlimit(2), calloc(3), and malloc(3) in the *Programmer's Reference Manual.*

Stephenson, C.J., *Fast Fits*, in *Proceedings of the ACM 9th Symposium on Operating Systems, SIGOPS Operating Systems Review*, vol. 17, no. 5, October 1983.

Core Wars, in *Scientific American*, May 1984.

NOTES

alloca is machine-, compiler-, and most of all, system-dependent. Its use is strongly discouraged.

NAME

bstring: bcopy, bcmp, bzero, ffs – bit and byte string operations

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
bcopy(b1, b2, length)
char *b1, *b2;
int length;

int bcmp(b1, b2, length)
char *b1, *b2;
int length;

bzero(b, length)
char *b;
int length;
```

DESCRIPTION

The functions bcopy, bcmp, and bzero operate on variable length strings of bytes. They do not check for null bytes as the routines in string(3) do.

bcopy copies *length* bytes from string *b1* to the string *b2*. Overlapping strings are handled correctly.

bcmp compares byte string *b1* against byte string *b2*, returning zero if they are identical, 1 otherwise. Both strings are assumed to be *length* bytes long. bcmp of length zero bytes always returns zero.

bzero places *length* 0 bytes in the string *b*.

CAVEAT

The bcmp and bcopy routines take parameters backwards from strcmp and strcpy.

SEE ALSO

string(3C) in the *Programmer's Reference Manual.*

NAME

dbm: dbminit, dbmclose, fetch, store, delete, firstkey, nextkey – data base subroutines

SYNOPSIS

```
cc [ flag... ] file ... -ldbm

#include <dbm.h>

typedef struct {
char *dptr;
int dsize;
} datum;

dbminit(file)
char *file;

dbmclose

datum fetch(key)
datum key;

store(key, content)
datum key, content;

delete(key)
datum key;

datum firstkey

datum nextkey(key)
datum key;
```

DESCRIPTION

Note: the dbm library has been superceded by ndbm(3), and is now implemented using ndbm.

These functions maintain key/content pairs in a data base. The functions will handle very large (a billion blocks) databases and will access a keyed item in one or two file system accesses. The functions are obtained with the loader option -libdbm.

keys and *contents* are described by the datum typedef. A datum specifies a string of *dsize* bytes pointed to by *dptr*. Arbitrary binary data, as well as normal ASCII strings, are allowed. The data base is stored in two files. One file is a directory containing a bit map and has .dir as its suffix. The second file contains all data and has .pag as its suffix.

Before a database can be accessed, it must be opened by dbminit. At the time of this call, the files *file*.dir and *file*.pag must exist. An empty database is created by creating zero-length .dir and .pag files.

A database may be closed by calling dbmclose. You must close a database before opening a new one.

Once open, the data stored under a key is accessed by fetch and data is placed under a key by store. A key (and its associated contents) is deleted by delete. A linear pass through all keys in a database may be made, in an (apparently) random order, by use of firstkey and nextkey. firstkey will return the first key

in the database. With any key `nextkey` will return the next key in the database. This code will traverse the data base:

```
for (key = firstkey; key.dptr != NULL; key = nextkey(key))
```

SEE ALSO

ndbm(3).

RETURN VALUE

All functions that return an `int` indicate errors with negative values. A zero return indicates no error. Routines that return a `datum` indicate errors with a NULL (0) *dptr*.

NOTES

The `.pag` file will contain holes so that its apparent size is about four times its actual content. Older versions of the UNIX operating system may create real file blocks for these holes when touched. These files cannot be copied by normal means (`cp`(1), `cat`(1), `tar`(1), `ar`(1)) without filling in the holes.

dptr pointers returned by these subroutines point into static storage that is changed by subsequent calls.

The sum of the sizes of a key/content pair must not exceed the internal block size (currently 1024 bytes). Moreover all key/content pairs that hash together must fit on a single block. `store` will return an error in the event that a disk block fills with inseparable data.

`delete` does not physically reclaim file space, although it does make it available for reuse.

The order of keys presented by `firstkey` and `nextkey` depends on a hashing function, not on anything interesting.

There are no interlocks and no reliable cache flushing; thus concurrent updating and reading is risky.

NAME

decimal_to_floating: decimal_to_single, decimal_to_double, decimal_to_extended – convert decimal record to floating-point value

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <floatingpoint.h>

void decimal_to_single(px, pm, pd, ps)
single *px ;
decimal_mode *pm;
decimal_record *pd;
fp_exception_field_type *ps;

void decimal_to_double(px, pm, pd, ps)
double *px ;
decimal_mode *pm;
decimal_record *pd;
fp_exception_field_type *ps;

void decimal_to_extended(px, pm, pd, ps)
extended *px ;
decimal_mode *pm;
decimal_record *pd;
fp_exception_field_type *ps;
```

DESCRIPTION

The decimal_to_floating functions convert the decimal record at **pd* into a floating-point value at **px*, observing the modes specified in **pm* and setting exceptions in **ps*. If there are no IEEE exceptions, **ps* will be zero.

pd->sign and *pd->fpclass* are always taken into account. *pd->exponent* and *pd->ds* are used when *pd->fpclass* is *fp_normal* or *fp_subnormal*. In these cases *pd->ds* must contain one or more ASCII digits followed by a NULL. **px* is set to a correctly rounded approximation to

```
(pd->sign) * (pd->ds) *10** (pd->exponent)
```

Thus if *pd->exponent* == −2 and *pd->ds* == "1234", **px* will get 12.34 rounded to storage precision. *pd->ds* cannot have more than DECIMAL_STRING_LENGTH−1 significant digits because one character is used to terminate the string with a NULL. If *pd->more!=0* on input then additional nonzero digits follow those in *pd->ds*; *fp_inexact* is set accordingly on output in **ps*.

**px* is correctly rounded according to the IEEE rounding modes in *pm->rd*. **ps* is set to contain *fp_inexact*, *fp_underflow*, or *fp_overflow* if any of these arise.

pd->ndigits, *pm->df*, and *pm->ndigits* are not used.

strtod(3C), scanf(3S), fscanf(), and sscanf() all use decimal_to_double.

SEE ALSO

scanf(3S), strtod(3C) in the *Programmer's Reference Manual.*

NAME

econvert, fconvert, gconvert, seconvert, sfconvert, sgconvert – output conversion

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <floatingpoint.h>

char *econvert(value, ndigit, decpt, sign, buf)
double value;
int ndigit, *decpt, *sign;
char *buf;

char *fconvert(value, ndigit, decpt, sign, buf)
double value;
int ndigit, *decpt, *sign;
char *buf;

char *gconvert(value, ndigit, trailing, buf)
double value;
int ndigit;
int trailing;
char *buf;

char *seconvert(value, ndigit, decpt, sign, buf)
single *value;
int ndigit, *decpt, *sign;
char *buf;

char *sfconvert(value, ndigit, decpt, sign, buf)
single *value;
int ndigit, *decpt, *sign;
char *buf;

char *sgconvert(value, ndigit, trailing, buf)
single *value;
int ndigit;
int trailing;
char *buf;
```

DESCRIPTION

econvert converts the *value* to a NULL-terminated string of *ndigit* ASCII digits in *buf* and returns a pointer to *buf*. *buf* should contain at least *ndigit*+1 characters. The position of the decimal point relative to the beginning of the string is stored indirectly through *decpt*. Thus *buf* == "314" and **decpt* == 1 corresponds to the numerical value 3.14, while *buf* == "314" and **decpt* == −1 corresponds to the numerical value .0314. If the sign of the result is negative, the word pointed to by *sign* is nonzero; otherwise it is zero. The least significant digit is rounded.

fconvert works much like econvert, except that the correct digit has been rounded as if for sprintf(%w.nf) output with *n*=*ndigit* digits to the right of the decimal point. *ndigit* can be negative to indicate rounding to the left of the decimal point. The return value is a pointer to *buf*. *buf* should contain at least *310+max*(0,*ndigit*) characters to accomodate any double-precision *value*.

`gconvert` converts the *value* to a `NULL`-terminated ASCII string in *buf* and returns a pointer to *buf*. It produces *ndigit* significant digits in fixed-decimal format, like `sprintf(%w.nf)`, if possible, and otherwise in floating-decimal format, like `sprintf(%w.ne)`; in either case *buf* is ready for printing, with sign and exponent. The result corresponds to that obtained by

```
(void) sprintf(buf,''%w.ng'',value) ;
```

If *trailing*= 0, trailing zeros and a trailing point are suppressed, as in `sprintf(%g)`. If *trailing*!= 0, trailing zeros and a trailing point are retained, as in `sprintf(%#g)`.

`seconvert`, `sfconvert`, and `sgconvert` are single-precision versions of these functions, and are more efficient than the corresponding double-precision versions. A pointer rather than the value itself is passed to avoid C's usual conversion of single-precision arguments to double.

IEEE Infinities and NaNs are treated similarly by these functions. "NaN" is returned for NaN, and "Inf" or "Infinity" for Infinity. The longer form is produced when *ndigit* ≥ 8.

SEE ALSO

`sprintf`(3S) in the *Programmer's Reference Manual.*

NAME

floating_to_decimal: single_to_decimal, double_to_decimal, extended_to_decimal - convert floating-point value to decimal record

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
#include <floatingpoint.h>

void single_to_decimal(px, pm, pd, ps)
single *px ;
decimal_mode *pm;
decimal_record *pd;
fp_exception_field_type *ps;

void double_to_decimal(px, pm, pd, ps)
double *px ;
decimal_mode *pm;
decimal_record *pd;
fp_exception_field_type *ps;

void extended_to_decimal(px, pm, pd, ps)
extended *px ;
decimal_mode *pm;
decimal_record *pd;
fp_exception_field_type *ps;
```

DESCRIPTION

The floating_to_decimal functions convert the floating-point value at **px* into a decimal record at **pd*, observing the modes specified in **pm* and setting exceptions in **ps*. If there are no IEEE exceptions, **ps* will be zero.

If **px* is zero, infinity, or NaN, then only *pd->sign* and *pd->fpclass* are set. Otherwise *pd->exponent* and *pd->ds* are also set so that

```
(pd->sign) * (pd->ds) *10** (pd->exponent)
```

is a correctly rounded approximation to **px*. *pd->ds* has at least one and no more than DECIMAL_STRING_LENGTH-1 significant digits because one character is used to terminate the string with a NULL.

pd->ds is correctly rounded according to the IEEE rounding modes in *pm->rd*. **ps* has *fp_inexact* set if the result was inexact, and has *fp_overflow* set if the string result does not fit in *pd->ds* because of the limitation DECIMAL_STRING_LENGTH.

If *pm->df==floating_form*, then *pd->ds* always contains *pm->ndigits* significant digits. Thus if **px* == 12.34 and *pm->ndigits* == 8, then *pd->ds* will contain 12340000 and *pd->exponent* will contain -6.

If *pm->df==fixed_form* and *pm->ndigits* >= 0, then *pd->ds* always contains *pm->ndigits* after the point and as many digits as necessary before the point. Since the latter is not known in advance, the total number of digits required is returned in *pd->ndigits*; if that number >= DECIMAL_STRING_LENGTH, then *ds* is undefined. *pd->exponent* always gets *-pm->ndigits*. Thus if **px* == 12.34 and *pm->ndigits* == 1, then *pd->ds* gets 123, *pd->exponent* gets -1, and *pd->ndigits* gets 3.

If *pm->df==fixed_form* and *pm->ndigits* < 0, then *pm->ds* always contains *-pm->ndigits* trailing zeros; in other words, rounding occurs *-pm->ndigits* to the left of the decimal point, but the digits rounded away are retained as zeros. The total number of digits required is in *pd->ndigits*. *pd->exponent* always gets 0. Thus if **px* == 12.34 and *pm->ndigits* == -1, then *pd->ds* gets 10, *pd->exponent* gets 0, and *pd->ndigits* gets 2.

pd->more is not used.

`econvert`(3), `fconvert`, `gconvert`, `printf`(3S), and `sprintf`, all use `double_to_decimal`.

SEE ALSO

`econvert`(3).
`printf`(3S) in the *Programmer's Reference Manual.*

NAME

floatingpoint – IEEE floating point definitions

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
#include <sys/ieeefp.h>
#include <fp.h>
```

DESCRIPTION

This file defines constants, types, variables, and functions used to implement standard floating point according to ANSI/IEEE Std 754-1985. The variables and functions are implemented in libucb.a. The included file <sys/ieeefp.h> defines certain types of interest to the kernel.

IEEE Rounding Modes:

fp_direction_type — The type of the IEEE rounding direction mode. Note: the order of enumeration varies according to hardware.

fp_direction — The IEEE rounding direction mode currently in force. This is a global variable that is intended to reflect the hardware state, so it should only be written indirectly through a function that also sets the hardware state.

fp_precision_type — The type of the IEEE rounding precision mode, which only applies on systems that support extended precision.

fp_precision — The IEEE rounding precision mode currently in force. This is a global variable that is intended to reflect the hardware state on systems with extended precision, so it should only be written indirectly.

SIGFPE handling:

sigfpe_code_type — The type of a SIGFPE code.

sigfpe_handler_type — The type of a user-definable SIGFPE exception handler called to handle a particular SIGFPE code.

SIGFPE_DEFAULT — A macro indicating the default SIGFPE exception handling, namely to perform the exception handling specified by calls to ieee_handler(3M), if any, and otherwise to dump core using abort(3).

SIGFPE_IGNORE — A macro indicating an alternate SIGFPE exception handling, namely to ignore and continue execution.

SIGFPE_ABORT — A macro indicating an alternate SIGFPE exception handling, namely to abort with a core dump.

IEEE Exception Handling:

N_IEEE_EXCEPTION — The number of distinct IEEE floating-point exceptions.

`fp_exception_type` The type of the `N_IEEE_EXCEPTION` exceptions. Each exception is given a bit number.

`fp_exception_field_type`
The type intended to hold at least `N_IEEE_EXCEPTION` bits corresponding to the IEEE exceptions numbered by `fp_exception_type`. Thus `fp_inexact` corresponds to the least significant bit and `fp_invalid` to the fifth least significant bit. Note: some operations may set more than one exception.

`fp_accrued_exceptions`
The IEEE exceptions between the time this global variable was last cleared, and the last time a function was called to update the variable by obtaining the hardware state.

`ieee_handlers` An array of user-specifiable signal handlers for use by the standard `SIGFPE` handler for IEEE arithmetic-related `SIGFPE` codes. Since IEEE trapping modes correspond to hardware modes, elements of this array should only be modified with a function like `ieee_handler`(3M) that performs the appropriate hardware mode update. If no `sigfpe_handler` has been declared for a particular IEEE-related `SIGFPE` code, then the related `ieee_handlers` will be invoked.

IEEE Formats and Classification:

single;extended Definitions of IEEE formats.

`fp_class_type` An enumeration of the various classes of IEEE values and symbols.

IEEE Base Conversion:

The functions described under `floating_to_decimal`(3) and `decimal_to_floating`(3) not only satisfy the IEEE Standard, but also the stricter requirements of correct rounding for all arguments.

`DECIMAL_STRING_LENGTH`
The length of a `decimal_string`.

`decimal_string` The digit buffer in a `decimal_record`.

`decimal_record` The canonical form for representing an unpacked decimal floating-point number.

`decimal_form` The type used to specify fixed or floating binary to decimal conversion.

`decimal_mode` A struct that contains specifications for conversion between binary and decimal.

`decimal_string_form`
An enumeration of possible valid character strings representing floating-point numbers, infinities, or NaNs.

FILES

/usr/include/sys/ieeefp.h
/usr/include/fp.h
/usr/ucblib/libucb.a

SEE ALSO

decimal_to_floating(3), econvert(3), floating_to_decimal(3), ieee_handler(3M), sigfpe(3).

abort(3), strtod(3) in the *Programmer's Reference Manual.*

NAME

fopen, freopen, fdopen – open a stream

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <stdio.h>

FILE *fopen(filename, type)
char *filename, *type;

FILE *freopen(filename, type, stream)
char *filename, *type;
FILE *stream;

FILE *fdopen(fildes, type)
int fildes;
char *type;
```

DESCRIPTION

fopen opens the file named by *filename* and associates a stream with it. If the open succeeds, fopen returns a pointer to be used to identify the stream in subsequent operations.

filename points to a character string that contains the name of the file to be opened.

type is a character string having one of the following values:

- r open for reading
- w truncate or create for writing
- a append: open for writing at end of file, or create for writing
- r+ open for update (reading and writing)
- w+ truncate or create for update
- a+ append; open or create for update at EOF

freopen opens the file named by *filename* and associates the stream pointed to by *stream* with it. The *type* argument is used just as in fopen. The original stream is closed, regardless of whether the open ultimately succeeds. If the open succeeds, freopen returns the original value of *stream*.

freopen is typically used to attach the preopened streams associated with stdin, stdout, and stderr to other files.

fdopen associates a stream with the file descriptor *fildes*. File descriptors are obtained from calls like open, dup, creat, or pipe(2), which open files but do not return streams. Streams are necessary input for many of the Section 3S library routines. The *type* of the stream must agree with the mode of the open file.

When a file is opened for update, both input and output may be done on the resulting stream. However, output may not be directly followed by input without an intervening fseek or rewind, and input may not be directly followed by output without an intervening fseek, rewind, or an input operation which encounters EOF.

SEE ALSO

open(2), pipe(2), fclose(3S), fseek(3S), fopen(3S), malloc(3C) in the *Programmer's Reference Manual.*

RETURN VALUE

fopen, freopen, and fdopen return a NULL pointer on failure.

NOTES

fopen differs from the library routine of the same name in the base system only in interface.

In order to support the same number of open files that the system does, fopen must allocate additional memory for data structures using calloc [see malloc(3)] after 64 files have been opened. This confuses some programs which use their own memory allocators.

NAME

ftime – get date and time

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
#include <sys/types.h>
#include <sys/timeb.h>

ftime(tp)
struct timeb *tp;
```

DESCRIPTION

The ftime entry fills in a structure pointed to by its argument, as defined by <sys/timeb.h>:

```
struct timeb
{
        time_t  time;
        unsigned short millitm;
        short   timezone;
        short   dstflag;
};
```

The structure contains the time since the epoch in seconds, up to 1000 milliseconds of more-precise interval, the local time zone (measured in minutes of time westward from Greenwich), and a flag that, if nonzero, indicates that Daylight Saving time applies locally during the appropriate part of the year.

SEE ALSO

date(1) in the *User's Reference Manual.*
gettimeofday(2), ctime(3) in the *Programmer's Reference Manual.*

NAME

getdtablesize – get descriptor table size

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
long getdtablesize()
```

DESCRIPTION

Each process has a descriptor table which is guaranteed to have at least 20 slots. The entries in the descriptor table are numbered with small integers starting at 0. The call `getdtablesize` returns the current maximum size of this table by calling the `getrlimit` system call.

SEE ALSO

close(2), dup(2), getrlimit(2), and open(2) in the *Programmer's Reference Manual.*

NAME

gethostid – get unique identifier of current host

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
gethostid()
```

DESCRIPTION

gethostid returns the 32-bit identifier for the current host, which should be unique across all hosts. This number is usually taken from the CPU board's ID PROM.

This routine resides in libucb.

SEE ALSO

hostid(1)
sysinfo(2) in the *Programmer's Reference Manual.*

NAME

gethostname, sethostname – get/set name of current host

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
int gethostname(name, namelen)
char *name;
int namelen;

int sethostname(name, namelen)
char *name;
int namelen;
```

DESCRIPTION

gethostname returns the standard host name for the current processor, as previously set by sethostname. The parameter *namelen* specifies the size of the array pointed to by *name*. The returned name is null-terminated unless insufficient space is provided.

sethostname sets the name of the host machine to be *name*, which has length *namelen*. This call is restricted to the privileged user and is normally used only when the system is bootstrapped.

RETURN VALUE

If the call succeeds a value of 0 is returned. If the call fails, then a value of −1 is returned and an error code is placed in the global location errno.

ERRORS

The following error may be returned by these calls:

EFAULT	The *name* or *namelen* parameter gave an invalid address.
EPERM	The caller was not the privileged user. Note: this error only applies to sethostname.

SEE ALSO

gethostid(3)

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

NOTES

Host names are limited to MAXHOSTNAMELEN characters, currently 256. (See the param.h header file.)

NAME

getpagesize – get system page size

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
int getpagesize(VOID);
```

DESCRIPTION

getpagesize returns the number of bytes in a page. Page granularity is the granularity of many of the memory management calls.

The page size is a system page size and need not be the same as the underlying hardware page size.

SEE ALSO

pagesize(1)
brk(2) in the *Programmer's Reference Manual.*

NAME

getpriority, setpriority – get/set program scheduling priority

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
#include <sys/time.h>
#include <sys/resource.h>

int getpriority(which, who)
int which, who;

int setpriority(which, who, prio)
int which, who, prio;
```

DESCRIPTION

The scheduling priority of the process, process group, or user, as indicated by *which* and *who* is obtained with getpriority and set with setpriority The default priority is 0; lower priorities cause more favorable scheduling.

which is one of PRIO_PROCESS, PRIO_PGRP, or PRIO_USER, and *who* is interpreted relative to *which* (a process identifier for PRIO_PROCESS, process group identifier for PRIO_PGRP, and a user ID for PRIO_USER). A zero value of *who* denotes the current process, process group, or user.

getpriority returns the highest priority (lowest numerical value) enjoyed by any of the specified processes. setpriority sets the priorities of all of the specified processes to the value specified by *prio*. If *prio* is less than −20, a value of −20 is used; if it is greater than 20, a value of 20 is used. Only the privileged user may lower priorities.

RETURN VALUE

Since getpriority can legitimately return the value −1, it is necessary to clear the external variable errno prior to the call, then check it afterward to determine if a −1 is an error or a legitimate value. The setpriority call returns 0 if there is no error, or −1 if there is.

ERRORS

getpriority and setpriority may return one of the following errors:

ESRCH No process was located using the *which* and *who* values specified.

EINVAL *which* was not one of PRIO_PROCESS, PRIO_PGRP, or PRIO_USER.

In addition to the errors indicated above, setpriority may fail with one of the following errors returned:

EPERM A process was located, but one of the following is true:

- Neither its effective nor real user ID matched the effective user ID of the caller, and neither the effective nor the real user ID of the process executing the setpriority was the privileged user.
- The call to getpriority would have changed a process' priority to a value lower than its current value, and the effective user ID of the process executing the call was not that of the privileged user.

SEE ALSO

`renice`(1M).

`nice`(1) in the *User's Reference Manual.*
`fork`(2) in the *Programmer's Reference Manual.*

NOTES

It is not possible for the process executing `setpriority` to lower any other process down to its current priority, without requiring privileged user privileges.

NAME

getrusage – get information about resource utilization

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
#include <sys/time.h>
#include <sys/resource.h>

getrusage(who, rusage)
int who;
struct rusage *rusage;
```

DESCRIPTION

getrusage returns information about the resources utilized by the current process, or all its terminated child processes. The interpretation for some values reported, such as ru_idrss, are dependent on the clock tick interval. This interval is an implementation dependent value.

The who parameter is one of RUSAGE_SELF or RUSAGE_CHILDREN. The buffer to which *rusage* points will be filled in with the following structure:

```
struct     rusage {
    struct timeval ru_utime;    /* user time used */
    struct timeval ru_stime;    /* system time used */
    int    ru_maxrss;           /* maximum resident set size */
    int    ru_ixrss;            /* currently 0 */
    int    ru_idrss;            /* integral resident set size */
    int    ru_isrss;            /* currently 0 */
    int    ru_minflt;           /* page faults not requiring physical I/O */
    int    ru_majflt;           /* page faults requiring physical I/O */
    int    ru_nswap;            /* swaps */
    int    ru_inblock;          /* block input operations */
    int    ru_oublock;          /* block output operations */
    int    ru_msgsnd;           /* messages sent */
    int    ru_msgrcv;           /* messages received */
    int    ru_nsignals;         /* signals received */
    int    ru_nvcsw;            /* voluntary context switches */
    int    ru_nivcsw;           /* involuntary context switches */
};
```

The fields are interpreted as follows:

ru_utime — The total amount of time spent executing in user mode. Time is given in seconds and microseconds.

ru_stime — The total amount of time spent executing in system mode. Time is given in seconds and microseconds.

ru_maxrss — The maximum resident set size. Size is given in pages (the size of a page, in bytes, is given by the getpagesize(3) system call). Also, see NOTES.

`ru_ixrss`	Currently returns 0.
`ru_idrss`	An integral value indicating the amount of memory in use by a process while the process is running. This value is the sum of the resident set sizes of the process running when a clock tick occurs. The value is given in pages times clock ticks. Note: it does not take sharing into account. Also, see NOTES.
`ru_isrss`	Currently returns 0.
`ru_minflt`	The number of page faults serviced which did not require any physical I/O activity. Also, see NOTES.
`ru_majflt`	The number of page faults serviced which required physical I/O activity. This could include page ahead operations by the kernel. Also, see NOTES.
`ru_nswap`	The number of times a process was swapped out of main memory.
`ru_inblock`	The number of times the file system had to perform input in servicing a `read`(2) request.
`ru_oublock`	The number of times the file system had to perform output in servicing a `write`(2) request.
`ru_msgsnd`	The number of messages sent over sockets.
`ru_msgrcv`	The number of messages received from sockets.
`ru_nsignals`	The number of signals delivered.
`ru_nvcsw`	The number of times a context switch resulted due to a process voluntarily giving up the processor before its time slice was completed (usually to await availability of a resource).
`ru_nivcsw`	The number of times a context switch resulted due to a higher priority process becoming runnable or because the current process exceeded its time slice.

RETURN VALUE

If successful, the value of the appropriate structure is filled in, and 0 is returned. If the call fails, a −1 is returned.

ERRORS

`getrusage` will fail if:

`EINVAL`	The `who` parameter is not a valid value.
`EFAULT`	The address specified by the `rusage` argument is not in a valid portion of the process's address space.

SEE ALSO

`sar`(1M) in the *System Administrator's Reference Manual.*
`gettimeofday`(3), `read`(2), `times`(2), `wait`(3), `write`(2) in the *Programmer's Reference Manual.*

NOTES

Only the *timeval* fields of `struct rusage` are supported in this implementation.

The numbers `ru_inblock` and `ru_oublock` account only for real I/O, and are approximate measures at best. Data supplied by the caching mechanism is charged only to the first process to read and the last process to write the data.

The way resident set size is calculated is an approximation, and could misrepresent the true resident set size.

Page faults can be generated from a variety of sources and for a variety of reasons. The customary cause for a page fault is a direct reference by the program to a page which is not in memory. Now, however, the kernel can generate page faults on behalf of the user, for example, servicing `read`(2) and `write`(2) system calls. Also, a page fault can be caused by an absent hardware translation to a page, even though the page is in physical memory.

In addition to hardware detected page faults, the kernel may cause pseudo page faults in order to perform some housekeeping. For example, the kernel may generate page faults, even if the pages exist in physical memory, in order to lock down pages involved in a raw I/O request.

By definition, *major* page faults require physical I/O, while *minor* page faults do not require physical I/O. For example, reclaiming the page from the free list would avoid I/O and generate a minor page fault. More commonly, minor page faults occur during process startup as references to pages which are already in memory. For example, if an address space faults on some hot executable or shared library, this results in a minor page fault for the address space. Also, any one doing a `read`(2) or `write`(2) to something that is in the page cache will get a minor page fault(s) as well.

There is no way to obtain information about a child process which has not yet terminated.

NAME

gettimeofday, settimeofday – get or set the date and time

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
#include <sys/time.h>

int gettimeofday(tp, tzp)
struct timeval *tp;
struct timezone *tzp;       /* obsolete */

int settimeofday(tp, tzp)
struct timeval *tp;
struct timezone *tzp;       /* obsolete */
```

DESCRIPTION

The system's notion of the current Greenwich time is obtained with the gettimeofday call, and set with the settimeofday call. The current time is expressed in elapsed seconds and microseconds since 00:00 GMT, January 1, 1970 (zero hour). The resolution of the system clock is hardware dependent; the time may be updated continuously, or in "ticks."

tp points to a timeval structure, which includes the following members:

```
long tv_sec;  /* seconds since Jan. 1, 1970 */
long tv_usec; /* and microseconds */
```

If *tp* is a NULL pointer, the current time information is not returned or set.

tzp is an obsolete pointer formerly used to get and set timezone information. *tzp* is now ignored. Timezone information is now handled using the TZ environment variable; see timezone(4).

Only the privileged user may set the time of day.

RETURN VALUE

A −1 return value indicates an error occurred; in this case an error code is stored in the global variable errno.

ERRORS

The following error codes may be set in errno:

EINVAL	*tp* specifies an invalid time.
EPERM	A user other than the privileged user attempted to set the time.

SEE ALSO

date(1) in the *User's Reference Manual.*
adjtime(2), ctime(3C), gettimeofday(3C), timezone(4) in the *Programmer's Reference Manual.*

NOTES

Time is never correct enough to believe the microsecond values.

tzp is ignored.

NAME

getusershell, setusershell, endusershell – get legal user shells

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
char *getusershell()

setusershell()
endusershell()
```

DESCRIPTION

getusershell returns a pointer to a legal user shell as defined by the system manager in the file /etc/shells. If /etc/shells does not exist, the locations of the standard system shells, /usr/bin/csh, /usr/bin/sh, and /usr/bin/ksh are returned.

getusershell reads the next line (opening the file if necessary); setusershell rewinds the file; endusershell closes it.

FILES

/etc/shells
/usr/bin/csh
/usr/bin/ksh
/usr/bin/sh

RETURN VALUE

The routine getusershell returns a NULL pointer (0) on EOF or error.

NOTES

All information is contained in a static area so it must be copied if it is to be saved.

NAME

getwd – get current working directory pathname

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <sys/param.h>

char *getwd(pathname)
char pathname[MAXPATHLEN];
```

DESCRIPTION

getwd copies the absolute pathname of the current working directory to *pathname* and returns a pointer to the result.

RETURN VALUE

getwd returns zero and places a message in *pathname* if an error occurs.

SEE ALSO

getcwd(3C) in the *Programmer's Reference Manual.*

NAME

ieee_functions, fp_class, isnan, copysign, scalbn – miscellaneous functions for IEEE arithmetic

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <fp.h>
#include <math.h>
#include <stdio.h>

enum fp_class_type fp_class(x)
double x;

int isnan(x)
double x;

double copysign(x,y)
double x, y;

double scalbn(x,n)
double x; int n;
```

DESCRIPTION

Most of these functions provide capabilities required by ANSI/IEEE Std 754-1985 or suggested in its appendix.

`fp_class(`*x*`)` corresponds to the IEEE's class() and classifies *x* as zero, subnormal, normal, ∞, or quiet or signaling *NaN*; `/usr/ucbinclude/sys/ieeefp.h` defines `enum fp_class_type`. The following function returns 0 if the indicated condition is not satisfied:

`isnan(`*x*`)` returns 1 if *x* is *NaN*

`copysign(`*x*`,`*y*`)` returns *x* with *y*'s sign bit.

`scalbn(`*x*`,`*n*`)` returns *x*`* 2**n` computed by exponent manipulation rather than by actually performing an exponentiation or a multiplication. Thus

`1 ≤ scalbn(fabs(`*x*`),-ilogb(`*x*`)) < 2`

for every *x* except 0, ∞, and *NaN*.

FILES

/usr/ucbinclude/sys/ieeefp.h
/usr/ucbinclude/math.h
/usr/include/values.h

NAME

ieee_handler – IEEE exception trap handler function

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <fp.h>

int ieee_handler(action,exception,hdl)
char action[], exception[];
sigfpe_handler_type hdl;
```

DESCRIPTION

This function provides easy exception handling to exploit ANSI/IEEE Std 754-1985 arithmetic in a C program. All arguments are pointers to strings. Results arising from invalid arguments and invalid combinations are undefined for efficiency.

There are three types of *action* : "get", "set", and "clear". There are five types of *exception* :

"inexact"	
"division"	... division by zero exception
"underflow"	
"overflow"	
"invalid"	
"all"	... all five exceptions above
"common"	... invalid, overflow, and division exceptions

Note: "all" and "common" only make sense with "set" or "clear".

hdl contains the address of a signal-handling routine. <fp.h> defines *sigfpe_handler_type*.

"get" will get the location of the current handler routine for *exception* in hdl . "set" will set the routine pointed at by hdl to be the handler routine and at the same time enable the trap on *exception*, except when hdl == SIGFPE_DEFAULT or SIGFPE_IGNORE; then ieee_handler will disable the trap on *exception*. When hdl == SIGFPE_ABORT, any trap on *exception* will dump core using abort(3). "clear" "all" disables trapping on all five exceptions.

Two steps are required to intercept an IEEE-related SIGFPE code with ieee_handler:

1) Set up a handler with ieee_handler.

2) Perform a floating-point operation that generates the intended IEEE exception.

Unlike sigfpe(3), ieee_handler also adjusts floating-point hardware mode bits affecting IEEE trapping. For "clear", "set" SIGFPE_DEFAULT, or "set" SIGFPE_IGNORE, the hardware trap is disabled. For any other "set", the hardware trap is enabled.

SIGFPE signals can be handled using sigvec(2), signal(3), signal(3F), sigfpe(3), or ieee_handler(3M). In a particular program, to avoid confusion, use only one of these interfaces to handle SIGFPE signals.

RETURN VALUE

ieee_handler normally returns 0. In the case of "set", 1 will be returned if the action is not available (for instance, not supported in hardware).

EXAMPLE

A user-specified signal handler might look like this:

```
void sample_handler( sig, code, scp, addr)
int sig ;                /* sig == SIGFPE always */
int code ;
struct sigcontext *scp ;
char *addr ;
{
     /*
        Sample user-written sigfpe code handler.
        Prints a message and continues.
        struct sigcontext is defined in <signal.h>.
      */
     printf("ieee exception code %x occurred at pc %X \n",
      code,scp->sc_pc);
}
```

and it might be set up like this:

```
extern void sample_handler;
main
{
     sigfpe_handler_type hdl, old_handler1, old_handler2;
/*
* save current overflow and invalid handlers
*/
     ieee_handler("get","overflow",old_handler1);
     ieee_handler("get","invalid", old_handler2);
/*
* set new overflow handler to sample_handler and set new
* invalid handler to SIGFPE_ABORT (abort on invalid)
*/
     hdl = (sigfpe_handler_type) sample_handler;
     if(ieee_handler("set","overflow",hdl) != 0)
          printf("ieee_handler can't set overflow \n");
     if(ieee_handler("set","invalid",SIGFPE_ABORT) != 0)
          printf("ieee_handler can't set invalid \n");
     ...
/*
* restore old overflow and invalid handlers
*/
     ieee_handler("set","overflow", old_handler1);
     ieee_handler("set","invalid", old_handler2);
}
```

FILES

`/usr/include/fp.h`
`/usr/include/signal.h`

SEE ALSO

`floatingpoint`(3), `ieee_handler`(3), `sigfpe`(3), `signal`(3) `sigvec`(3), `signal`(2), `abort`(3C) in the *Programmer's Reference Manual.*

NAME

index, rindex – string operations

SYNOPSIS

```
#include <string.h>

char *index(s, c)
char *s, c;

char *rindex(s, c)
char *s, c;
```

DESCRIPTION

These functions operate on NULL-terminated strings. They do not check for overflow of any receiving string.

index and rindex returns a pointer to the first (last) occurrence of character *c* in string *s*, or a NULL pointer if *c* does not occur in the string. The NULL character terminating a string is considered to be part of the string.

SEE ALSO

bstring(3), strings(3)

malloc(3C) in the *Programmer's Reference Manual.*

NOTES

For user convenience, these functions are declared in the optional <strings.h> header file.

On the Sun processor, as well as on many other machines, you can *not* use a NULL pointer to indicate a NULL string. A NULL pointer is an error and results in an abort of the program. If you wish to indicate a NULL string, you must have a pointer that points to an explicit NULL string. On some implementations of the C language on some machines, a NULL pointer, if dereferenced, would yield a NULL string; this highly non-portable trick was used in some programs. Programmers using a NULL pointer to represent an empty string should be aware of this portability issue; even on machines where dereferencing a NULL pointer does not cause an abort of the program, it does not necessarily yield a NULL string.

Character movement is performed differently in different implementations. Thus overlapping moves may yield surprises.

NAME

killpg – send signal to a process group

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
int killpg(pgrp, sig)
int pgrp, sig;
```

DESCRIPTION

killpg sends the signal *sig* to the process group *pgrp*. See sigvec(3) for a list of signals.

The real or effective user ID of the sending process must match the real or saved set-user ID of the receiving process, unless the effective user ID of the sending process is the privileged user. A single exception is the signal SIGCONT, which may always be sent to any descendant of the current process.

RETURN VALUE

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and the global variable errno is set to indicate the error.

ERRORS

killpg will fail and no signal will be sent if any of the following occur:

EINVAL	*sig* is not a valid signal number.
ESRCH	No processes were found in the specified process group.
EPERM	The effective user ID of the sending process is not privileged user, and neither its real nor effective user ID matches the real or saved set-user ID of one or more of the target processes.

SEE ALSO

sigvec(3)
kill(2), setpgrp(2), sigaction(2) in the *Programmer's Reference Manual.*

NAME

mctl – memory management control

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <sys/types.h>
#include <sys/mman.h>

mctl(caddr_t addr, size_t len, int function, void *arg);
```

DESCRIPTION

mctl applies a variety of control functions over pages identified by the mappings established for the address range [*addr, addr + len*). The function to be performed is identified by the argument *function*. Valid functions are defined in mman.h as follows.

MC_LOCK
: Lock the pages in the range in memory. This function is used to support mlock. See mlock(3) for semantics and usage. *arg* is ignored.

MC_LOCKAS
: Lock the pages in the address space in memory. This function is used to support mlockall. See mlockall(3) for semantics and usage. *addr* and *len* are ignored. *arg* is an integer built from the flags:

MCL_CURRENT	Lock current mappings
MCL_FUTURE	Lock future mappings

MC_SYNC
: Synchronize the pages in the range with their backing storage. Optionally invalidate cache copies. This function is used to support msync. See msync(3) for semantics and usage. *arg* is used to represent the *flags* argument to msync. It is constructed from an OR of the following values:

MS_SYNC	Synchronized write
MS_ASYNC	Return immediately
MS_INVALIDATE	Invalidate mappings

MS_ASYNC returns after all I/O operations are scheduled. MS_SYNC does not return until all I/O operations are complete. Specify exactly one of MS_ASYNC or MS_SYNC. MS_INVALIDATE invalidates all cached copies of data from memory, requiring them to be re-obtained from the object's permanent storage location upon the next reference.

MC_UNLOCK
: Unlock the pages in the range. This function is used to support munlock. See munlock(3) for semantics and usage. *arg* is ignored.

MC_UNLOCKAS
: Remove address space memory lock, and locks on all current mappings. This function is used to support munlockall(3). *addr* and *len* must have the value 0. *arg* is ignored.

RETURN VALUE

`mctl` returns 0 on success, −1 on failure.

ERRORS

`mctl` fails if:

`EAGAIN`	Some or all of the memory identified by the operation could not be locked due to insufficient system resources.
`EBUSY`	`MS_INVALIDATE` was specified and one or more of the pages is locked in memory.
`EINVAL`	*addr* is not a multiple of the page size as returned by `getpagesize`.
`EINVAL`	*addr* and/or *len* do not have the value 0 when `MC_LOCKAS` or `MC_UNLOCKAS` are specified.
`EINVAL`	*arg* is not valid for the function specified.
`EIO`	An I/O error occurred while reading from or writing to the file system.
`ENOMEM`	Addresses in the range [*addr, addr* + *len*) are invalid for the address space of a process, or specify one or more pages which are not mapped.
`EPERM`	The process's effective user ID is not super-user and one of `MC_LOCK`, `MC_LOCKAS`, `MC_UNLOCK`, or `MC_UNLOCKAS` was specified.

SEE ALSO

`getpagesize`(3).

`mmap`(2), `mlock`(3C), `mlockall`(3C), `msync`(3C) in the *Programmer's Reference Manual.*

NAME

mkstemp – make a unique file name

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
mkstemp(template)
char *template;
```

DESCRIPTION

mkstemp creates a unique file name, typically in a temporary filesystem, by replacing *template* with a unique file name, and returns a file descriptor for the template file open for reading and writing. The string in *template* should contain a file name with six trailing Xs; mkstemp replaces the Xs with a letter and the current process ID. The letter will be chosen so that the resulting name does not duplicate an existing file. mkstemp avoids the race between testing whether the file exists and opening it for use.

SEE ALSO

getpid(2), open(2), tmpfile(3S), tmpnam(3S) in the *Programmer's Reference Manual.*

RETURN VALUE

mkstemp returns −1 if no suitable file could be created.

NOTES

It is possible to run out of letters.

mkstemp actually changes the template string which you pass; this means that you cannot use the same template string more than once — you need a fresh template for every unique file you want to open.

When mkstemp is creating a new unique filename it checks for the prior existence of a file with that name. This means that if you are creating more than one unique filename, it is bad practice to use the same root template for multiple invocations of mkstemp.

NAME

mp: madd, msub, mult, mdiv, mcmp, min, mout, pow, gcd, rpow, msqrt, sdiv, itom, xtom, mtox, mfree – multiple precision integer arithmetic

SYNOPSIS

cc [*flag*...] *file* ... -lmp

```
#include <mp.h>

madd(a, b, c)
MINT *a, *b, *c;

msub(a, b, c)
MINT *a, *b, *c;

mult(a, b, c)
MINT *a, *b, *c;

mdiv(a, b, q, r)
MINT *a, *b, *q, *r;

mcmp(a,b)
MINT *a, *b;

min(a)
MINT *a;

mout(a)
MINT *a;

pow(a, b, c, d)
MINT *a, *b, *c, *d;

gcd(a, b, c)
MINT *a, *b, *c;

rpow(a, n, b)
MINT *a, *b;
short n;

msqrt(a, b, r)
MINT *a, *b, *r;

sdiv(a, n, q, r)
MINT *a, *q;
short n, *r;

MINT *itom(n)
short n;

MINT *xtom(s)
char *s;

char *mtox(a)
MINT *a;

void mfree(a)
MINT *a;
```

DESCRIPTION

These routines perform arithmetic on integers of arbitrary length. The integers are stored using the defined type MINT. Pointers to a MINT should be initialized using the function `itom`, which sets the initial value to *n*. Alternatively, `xtom` may be used to initialize a MINT from a string of hexadecimal digits. `mfree` may be used to release the storage allocated by the `itom` and `xtom` routines.

`madd`, `msub` and `mult` assign to their third arguments the sum, difference, and product, respectively, of their first two arguments. `mdiv` assigns the quotient and remainder, respectively, to its third and fourth arguments. `sdiv` is like `mdiv` except that the divisor is an ordinary integer. `msqrt` produces the square root and remainder of its first argument. `mcmp` compares the values of its arguments and returns 0 if the two values are equal, >0 if the first argument is greater than the second, and <0 if the second argument is greater than the first. `rpow` calculates *a* raised to the power *b*, while `pow` calculates this reduced modulo *m*. `min` and `mout` do decimal input and output. `gcd` finds the greatest common divisor of the first two arguments, returning it in the third argument. `mtox` provides the inverse of `xtom`. To release the storage allocated by `mtox`, use `free` [see `malloc`(3)].

Use the `-lbmp` loader option to obtain access to these functions.

RETURN VALUE

Illegal operations and running out of memory produce messages and core images.

FILES

`/usr/ucblib/libmp.a`

SEE ALSO

`malloc`(3) in the *Programmer's Reference Manual.*

NAME

ndbm: dbm_clearerr, dbm_close, dbm_delete, dbm_error, dbm_fetch, dbm_firstkey, dbm_nextkey, dbm_open, dbm_store – data base subroutines

SYNOPSIS

```
cc [ flag... ] file ... -ldbm

#include <ndbm.h>

typedef struct {
     char *dptr;
     int dsize;
} datum;

int dbm_clearerr(db)
DBM *db;

void dbm_close (db)
DBM *db;

int dbm_delete(db, key)
DBM *db;
datum key;

int dbm_error(db)
DBM *db;

datum dbm_fetch(db, key)
DBM *db;
datum key;

datum dbm_firstkey(db)
DBM *db;

datum dbm_nextkey(db)
DBM *db;

DBM *dbm_open(file, flags, mode)
char *file;
int flags, mode;

int dbm_store(db, key, content, flags)
DBM *db;
datum key, content;
int flags;
```

DESCRIPTION

These functions maintain *key/content* pairs in a data base. The functions will handle very large (a billion blocks) data base and will access a keyed item in one or two file system accesses. This package replaces the earlier dbm(3X) library, which managed only a single data base.

*key*s and *content*s are described by the datum typedef. A datum specifies a string of *dsize* bytes pointed to by *dptr*. Arbitrary binary data, as well as normal ASCII strings, are allowed. The data base is stored in two files. One file is a directory containing a bit map and has `.dir` as its suffix. The second file contains all data and has `.pag` as its suffix.

Before a data base can be accessed, it must be opened by dbm_open. This will open and/or create the files *file*.dir and *file*.pag depending on the flags parameter (see open(2V)).

A data base is closed by calling dbm_close.

Once open, the data stored under a key is accessed by dbm_fetch and data is placed under a key by dbm_store. The *flags* field can be either DBM_INSERT or DBM_REPLACE. DBM_INSERT will only insert new entries into the data base and will not change an existing entry with the same key. DBM_REPLACE will replace an existing entry if it has the same key. A key (and its associated contents) is deleted by dbm_delete. A linear pass through all keys in a data base may be made, in an (apparently) random order, by use of dbm_firstkey and dbm_nextkey. dbm_firstkey will return the first key in the data base. dbm_nextkey will return the next key in the data base. This code will traverse the data base:

```
for (key = dbm_firstkey(db); key.dptr != NULL; key = dbm_nextkey(db))
```

dbm_error returns non-zero when an error has occurred reading or writing the data base. dbm_clearerr resets the error condition on the named data base.

SEE ALSO

open(2), dbm(3X) in the *Programmer's Reference Manual.*

RETURN VALUE

All functions that return an int indicate errors with negative values. A zero return indicates no error. Routines that return a datum indicate errors with a NULL (0) *dptr*. If dbm_store is called with a *flags* value of DBM_INSERT and finds an existing entry with the same key, it returns 1.

NOTES

The .pag file will contain holes so that its apparent size is about four times its actual content. Older versions of the UNIX operating system may create real file blocks for these holes when touched. These files cannot be copied by normal means (cp(1), cat(1), tar(1), ar(1)) without filling in the holes.

dptr pointers returned by these subroutines point into static storage that is changed by subsequent calls.

The sum of the sizes of a *key/content* pair must not exceed the internal block size (currently 4096 bytes). Moreover all *key/content* pairs that hash together must fit on a single block. dbm_store will return an error in the event that a disk block fills with inseparable data.

dbm_delete does not physically reclaim file space, although it does make it available for reuse.

The order of keys presented by dbm_firstkey and dbm_nextkey depends on a hashing function.

There are no interlocks and no reliable cache flushing; thus concurrent updating and reading is risky.

NAME

nice – change priority of a process

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
int nice(incr)
int incr;
```

DESCRIPTION

The scheduling priority of the process is augmented by *incr*. Positive priorities get less service than normal. Priority 10 is recommended to users who wish to execute long-running programs without undue impact on system performance.

Negative increments are illegal, except when specified by the privileged user. The priority is limited to the range −20 (most urgent) to 20 (least). Requests for values above or below these limits result in the scheduling priority being set to the corresponding limit.

The priority of a process is passed to a child process by fork(2). For a privileged process to return to normal priority from an unknown state, nice should be called successively with arguments −40 (goes to priority −20 because of truncation), 20 (to get to 0), then 0 (to maintain compatibility with previous versions of this call).

RETURN VALUE

Upon successful completion, nice returns 0. Otherwise, a value of −1 is returned and errno is set to indicate the error.

ERRORS

The priority is not changed if:

EACCES The value of *incr* specified was negative, and the effective user ID is not the privileged user.

SEE ALSO

renice(1M)

nice(1), priocntl(2) in the *User's Reference Manual.*
fork(2), getpriority(2), priocntl(2) in the *Programmer's Reference Manual.*

NAME

nlist – get entries from symbol table

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
#include <nlist.h>

int nlist(filename, nl)
char *filename;
struct nlist *nl;
```

DESCRIPTION

nlist examines the symbol table from the executable image whose name is pointed to by *filename*, and selectively extracts a list of values and puts them in the array of nlist structures pointed to by *nl*. The name list pointed to by nl consists of an array of structures containing names, types and values. The n_name field of each such structure is taken to be a pointer to a character string representing a symbol name. The list is terminated by an entry with a NULL pointer (or a pointer to a NULL string) in the n_name field. For each entry in *nl*, if the named symbol is present in the executable image's symbol table, its value and type are placed in the n_value and n_type fields. If a symbol cannot be located, the corresponding n_type field of nl is set to zero.

RETURN VALUE

Upon normal completion, nlist returns the number of symbols that were not located in the symbol table. If an error occurs, nlist returns -1 and sets all of the n_type fields in members of the array pointed to by nl to zero.

SEE ALSO

a.out(4) in the *Programmer's Reference Manual.*

NAME

printf, fprintf, sprintf, vprintf, vfprintf, vsprintf – formatted output conversion

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
#include <stdio.h>
int printf(format [ , arg ] ... )
char *format;

int fprintf(stream, format [ , arg ] ... )
FILE *stream;
char *format;

char *sprintf(s, format [ , arg ] ... )
char *s, *format;

int vprintf(format, ap)
char *format;
va_list ap;

int vfprintf(stream, format, ap)
FILE *stream;
char *format;
va_list ap;

char *vsprintf(s, format, ap)
char *s, *format;
va_list ap;
```

DESCRIPTION

printf places output on the standard output stream stdout. fprintf places output on the named output *stream*. sprintf places "output," followed by the NULL character (\0), in consecutive bytes starting at *s; it is the user's responsibility to ensure that enough storage is available.

vprintf, vfprintf, and vsprintf are the same as printf, fprintf, and sprintf respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined by varargs(5).

Each of these functions converts, formats, and prints its *arg*s under control of the *format*. The *format* is a character string which contains two types of objects: plain characters, which are simply copied to the output stream, and conversion specifications, each of which causes conversion and printing of zero or more *arg*s. The results are undefined if there are insufficient *arg*s for the format. If the format is exhausted while *arg*s remain, the excess *arg*s are simply ignored.

Each conversion specification is introduced by the character %. After the %, the following appear in sequence:

> Zero or more *flags*, which modify the meaning of the conversion specification.

An optional decimal digit string specifying a minimum *field width*. If the converted value has fewer characters than the field width, it will be padded on the left (or right, if the left-adjustment flag '–', described below, has been given) to the field width. The padding is with blanks unless the field width digit string starts with a zero, in which case the padding is with zeros.

A *precision* that gives the minimum number of digits to appear for the d, i, o, u, x, or X conversions, the number of digits to appear after the decimal point for the e, E, and f conversions, the maximum number of significant digits for the g and G conversion, or the maximum number of characters to be printed from a string in s conversion. The precision takes the form of a period (.) followed by a decimal digit string; a NULL digit string is treated as zero. Padding specified by the precision overrides the padding specified by the field width.

An optional l (ell) specifying that a following d, i, o, u, x, or X conversion character applies to a long integer *arg*. An l before any other conversion character is ignored.

A character that indicates the type of conversion to be applied.

A field width or precision or both may be indicated by an asterisk (*) instead of a digit string. In this case, an integer *arg* supplies the field width or precision. The *arg* that is actually converted is not fetched until the conversion letter is seen, so the *arg*s specifying field width or precision must appear *before* the *arg* (if any) to be converted. A negative field width argument is taken as a '–' flag followed by a positive field width. If the precision argument is negative, it will be changed to zero.

The flag characters and their meanings are:

– The result of the conversion will be left-justified within the field.

+ The result of a signed conversion will always begin with a sign (+ or –).

blank If the first character of a signed conversion is not a sign, a blank will be prefixed to the result. This implies that if the blank and + flags both appear, the blank flag will be ignored.

This flag specifies that the value is to be converted to an "alternate form."For c, d, i, s, and u conversions, the flag has no effect. For o conversion, it increases the precision to force the first digit of the result to be a zero. For x or X conversion, a non-zero result will have 0x or 0X prefixed to it. For e, E, f, g, and G conversions, the result will always contain a decimal point, even if no digits follow the point (normally, a decimal point appears in the result of these conversions only if a digit follows it). For g and G conversions, trailing zeroes will *not* be removed from the result (which they normally are).

The conversion characters and their meanings are:

d,i,o,u,x,X

The integer *arg* is converted to signed decimal (d or i), unsigned octal (o), unsigned decimal (u), or unsigned hexadecimal notation (x and X), respectively; the letters abcdef are used for x conversion and the

letters ABCDEF for X conversion. The precision specifies the minimum number of digits to appear; if the value being converted can be represented in fewer digits, it will be expanded with leading zeroes. (For compatibility with older versions, padding with leading zeroes may alternatively be specified by prepending a zero to the field width. This does not imply an octal value for the field width.) The default precision is 1. The result of converting a zero value with a precision of zero is a NULL string.

f The float or double *arg* is converted to decimal notation in the style [−]*ddd.ddd* where the number of digits after the decimal point is equal to the precision specification. If the precision is missing, 6 digits are given; if the precision is explicitly 0, no digits and no decimal point are printed.

e,E The float or double *arg* is converted in the style [−]*d.ddde±ddd*, where there is one digit before the decimal point and the number of digits after it is equal to the precision; when the precision is missing, 6 digits are produced; if the precision is zero, no decimal point appears. The E format code will produce a number with E instead of e introducing the exponent. The exponent always contains at least two digits.

g,G The float or double *arg* is printed in style f or e (or in style E in the case of a G format code), with the precision specifying the number of significant digits. The style used depends on the value converted: style e or E will be used only if the exponent resulting from the conversion is less than −4 or greater than the precision. Trailing zeroes are removed from the result; a decimal point appears only if it is followed by a digit.

The e, E, f, g, and G formats print IEEE indeterminate values (infinity or not-a-number) as "Infinity" or "NaN" respectively.

c The character *arg* is printed.

s The *arg* is taken to be a string (character pointer) and characters from the string are printed until a NULL character (\0) is encountered or until the number of characters indicated by the precision specification is reached. If the precision is missing, it is taken to be infinite, so all characters up to the first NULL character are printed. A NULL value for *arg* will yield undefined results.

% Print a %; no argument is converted.

In no case does a non-existent or small field width cause truncation of a field; if the result of a conversion is wider than the field width, the field is simply expanded to contain the conversion result. Padding takes place only if the specified field width exceeds the actual width. Characters generated by printf and fprintf are printed as if putc(3S) had been called.

RETURN VALUE

Upon success, printf and fprintf return the number of characters transmitted, excluding the null character. vprintf and vfprintf return the number of characters transmitted. sprintf and vsprintf always return *s*. If an output error is encountered, printf, fprint, vprintf, and vfprintf, return EOF.

EXAMPLE

To print a date and time in the form "Sunday, July 3, 10:02," where *weekday* and *month* are pointers to NULL-terminated strings:

```
printf("%s, %s %i, %d:%.2d", weekday, month, day, hour, min);
```

To print π to 5 decimal places:

```
printf("pi = %.5f", 4 * atan(1. 0));
```

SEE ALSO

econvert(3)

putc(3S), scanf(3S), varargs(5), vprintf(3S) in the *Programmer's Reference Manual.*

NOTES

Very wide fields (>128 characters) fail.

NAME

psignal, sys_siglist – system signal messages

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
psignal(sig, s)
unsigned sig;
char *s;
```

```
char *sys_siglist[];
```

DESCRIPTION

psignal produces a short message on the standard error file describing the indicated signal. First the argument string *s* is printed, then a colon, then the name of the signal and a NEWLINE. Most usefully, the argument string is the name of the program which incurred the signal. The signal number should be from among those found in <signal.h>.

To simplify variant formatting of signal names, the vector of message strings sys_siglist is provided; the signal number can be used as an index in this table to get the signal name without the newline. The define NSIG defined in <signal.h> is the number of messages provided for in the table; it should be checked because new signals may be added to the system before they are added to the table.

SEE ALSO

signal(3)

perror(3C) in the *Programmer's Reference Manual.*

NAME

rand, srand – simple random number generator

SYNOPSIS

```
cc [ flag... ] file ... -lucb

srand(seed)
int seed;

rand()
```

DESCRIPTION

rand uses a multiplicative congruential random number generator with period 2^{32} to return successive pseudo-random numbers in the range from 0 to $2^{31}-1$.

srand can be called at any time to reset the random-number generator to a random starting point. The generator is initially seeded with a value of 1.

SEE ALSO

random(3).

drand48(2), drand(3C), rand(3C), srand(3C) in the *Programmer's Reference Manual.*

NOTES

The spectral properties of rand leave a great deal to be desired. drand48(2) and random(3) provide much better, though more elaborate, random-number generators.

The low bits of the numbers generated are not very random; use the middle bits. In particular the lowest bit alternates between 0 and 1.

NAME

random, srandom, initstate, setstate – better random number generator; routines for changing generators

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
long  random()

srandom(seed)
int  seed;

char  *initstate(seed, state, n)
unsigned  seed;
char  *state;
int  n;

char  *setstate(state)
char  *state;
```

DESCRIPTION

random uses a non-linear additive feedback random number generator employing a default table of size 31 long integers to return successive pseudo-random numbers in the range from 0 to $2^{31}-1$. The period of this random number generator is very large, approximately $16\times(2^{31}-1)$.

random/srandom have (almost) the same calling sequence and initialization properties as rand/srand [see rand(3C)]. The difference is that rand(3C) produces a much less random sequence—in fact, the low dozen bits generated by rand go through a cyclic pattern. All the bits generated by random are usable. For example,

```
random()&01
```

will produce a random binary value.

Unlike srand, srandom does not return the old seed because the amount of state information used is much more than a single word. Two other routines are provided to deal with restarting/changing random number generators. Like rand(3C), however, random will, by default, produce a sequence of numbers that can be duplicated by calling srandom with 1 as the seed.

The initstate routine allows a state array, passed in as an argument, to be initialized for future use. *n* specifies the size of *state* in bytes. initstate uses *n* to decide how sophisticated a random number generator it should use—the more state, the better the random numbers will be. Current "optimal" values for the amount of state information are 8, 32, 64, 128, and 256 bytes; other amounts will be rounded down to the nearest known amount. Using less than 8 bytes will cause an error. The seed for the initialization (which specifies a starting point for the random number sequence, and provides for restarting at the same point) is also an argument. initstate returns a pointer to the previous state information array.

Once a state has been initialized, the `setstate` routine provides for rapid switching between states. `setstate` returns a pointer to the previous state array; its argument state array is used for further random number generation until the next call to `initstate` or `setstate`.

Once a state array has been initialized, it may be restarted at a different point either by calling `initstate` (with the desired seed, the state array, and its size) or by calling both `setstate` (with the state array) and `srandom` (with the desired seed). The advantage of calling both `setstate` and `srandom` is that the size of the state array does not have to be remembered after it is initialized.

With 256 bytes of state information, the period of the random number generator is greater than 2^{69}, which should be sufficient for most purposes.

EXAMPLE

```
/* Initialize an array and pass it in to initstate. */
static long state1[32] = {
        3,
        0x9a319039, 0x32d9c024, 0x9b663182, 0x5da1f342,
        0x7449e56b, 0xbeb1dbb0, 0xab5c5918, 0x946554fd,
        0x8c2e680f, 0xeb3d799f, 0xb11ee0b7, 0x2d436b86,
        0xda672e2a, 0x1588ca88, 0xe369735d, 0x904f35f7,
        0xd7158fd6, 0x6fa6f051, 0x616e6b96, 0xac94efdc,
        0xde3b81e0, 0xdf0a6fb5, 0xf103bc02, 0x48f340fb,
        0x36413f93, 0xc622c298, 0xf5a42ab8, 0x8a88d77b,
        0xf5ad9d0e, 0x8999220b, 0x27fb47b9
        };
main()
{
        unsigned seed;
        int n;
        seed = 1;
        n = 128;
        initstate(seed, state1, n);
        setstate(state1);
        printf("%d0,random());
}
```

SEE ALSO

rand(3C).

drand48(2), drand(3C), rand(3C), srand(3C) in the *Programmer's Reference Manual.*

RETURN VALUE

If `initstate` is called with less than 8 bytes of state information, or if `setstate` detects that the state information has been garbled, error messages are printed on the standard error output.

NOTES

About two-thirds the speed of rand(3C).

NAME

reboot – reboot system or halt processor

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <sys/reboot.h>

reboot(howto, [ bootargs ] )
int howto;
char *bootargs;
```

DESCRIPTION

reboot reboots the system, and is invoked automatically in the event of unrecoverable system failures. *howto* is a mask of options passed to the bootstrap program. The system call interface permits only RB_HALT or RB_AUTOBOOT to be passed to the reboot program; the other flags are used in scripts stored on the console storage media, or used in manual bootstrap procedures. When none of these options (for instance RB_AUTOBOOT) is given, the system is rebooted from file /stand/unix. An automatic consistency check of the disks is then normally performed.

The bits of *howto* that are used are:

RB_HALT — the processor is simply halted; no reboot takes place. RB_HALT should be used with caution.

RB_ASKNAME — Interpreted by the bootstrap program itself, causing it to inquire as to what file should be booted. Normally, the system is booted from the file /stand/unix without asking.

RETURN VALUE

If successful, this call never returns. Otherwise, a −1 is returned and an error is returned in the global variable errno.

ERRORS

EPERM — The caller is not the super-user.

FILES

/vmunix

SEE ALSO

halt(1M) init(1M) reboot(1M)

intro(1M), crash(1M) in the *System Administrator's Reference Manual.*

NOTES

Any other *howto* argument causes /stand/unix to boot.

Only the super-user may reboot a machine.

NAME

regex, re_comp, re_exec – regular expression handler

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
char *re_comp(s)
char *s;

re_exec(s)
char *s;
```

DESCRIPTION

re_comp compiles a string into an internal form suitable for pattern matching. re_exec checks the argument string against the last string passed to re_comp.

re_comp returns a NULL pointer if the string *s* was compiled successfully; otherwise a string containing an error message is returned. If re_comp is passed 0 or a NULL string, it returns without changing the currently compiled regular expression.

re_exec returns 1 if the string *s* matches the last compiled regular expression, 0 if the string *s* failed to match the last compiled regular expression, and -1 if the compiled regular expression was invalid (indicating an internal error).

The strings passed to both re_comp and re_exec may have trailing or embedded NEWLINE characters; they are terminated by NULL characters. The regular expressions recognized are described in the manual entry for ed(1), given the above difference.

SEE ALSO

ed(1), ex(1), grep(1) in the *User's Reference Manual.*
regcmp(1), regexpr(3G), regcmp(3X), regexpr(5) in the *Programmer's Reference Manual.*

RETURN VALUE

re_exec returns -1 for an internal error.

re_comp returns one of the following strings if an error occurs:

```
No previous regular expression
Regular expression too long
unmatched \(
missing ]
too many \(\) pairs
unmatched \)
```

NAME

scandir, alphasort – scan a directory

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <sys/types.h>
#include <sys/dir.h>

scandir(dirname, &namelist, select, compar)
char *dirname;
struct direct **namelist;
int (*select)();
int (*compar)();

alphasort(d1, d2)
struct direct **d1, **d2;
```

DESCRIPTION

scandir reads the directory dirname and builds an array of pointers to directory entries using malloc(3C). The second parameter is a pointer to an array of structure pointers. The third parameter is a pointer to a routine which is called with a pointer to a directory entry and should return a non zero value if the directory entry should be included in the array. If this pointer is NULL, then all the directory entries will be included. The last argument is a pointer to a routine which is passed to qsort(3C) to sort the completed array. If this pointer is NULL, the array is not sorted. alphasort is a routine which will sort the array alphabetically.

scandir returns the number of entries in the array and a pointer to the array through the parameter *namelist.*

SEE ALSO

getdents(2), directory(3C), malloc(3C), qsort(3C) in the *Programmer's Reference Manual.*

RETURN VALUE

Returns –1 if the directory cannot be opened for reading or if malloc(3C) cannot allocate enough memory to hold all the data structures.

NAME

setbuf, setbuffer, setlinebuf, setvbuf – assign buffering to a stream

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <stdio.h>

setbuf(stream, buf)
FILE *stream;
char *buf;

setbuffer(stream, buf, size)
FILE *stream;
char *buf;
int size;

setlinebuf(stream)
FILE *stream;

int setvbuf(stream, buf, type, size)
FILE *stream;
char *buf;
int type, size;
```

DESCRIPTION

The three types of buffering available are unbuffered, block buffered, and line buffered. When an output stream is unbuffered, information appears on the destination file or terminal as soon as written; when it is block buffered many characters are saved up and written as a block; when it is line buffered characters are saved up until a NEWLINE is encountered or input is read from stdin. fflush (see fclose(3S)) may be used to force the block out early. Normally all files are block buffered. A buffer is obtained from malloc(3C) upon the first getc or putc(3S) on the file. If the standard stream stdout refers to a terminal it is line buffered. The standard stream stderr is unbuffered by default.

setbuf can be used after a stream has been opened but before it is read or written. It causes the array pointed to by *buf* to be used instead of an automatically allocated buffer. If *buf* is the NULL pointer, input/output will be completely unbuffered. A manifest constant BUFSIZ, defined in the <stdio.h> header file, tells how big an array is needed:

```
char buf[BUFSIZ];
```

setbuffer, an alternate form of setbuf, can be used after a stream has been opened but before it is read or written. It uses the character array *buf* whose size is determined by the *size* argument instead of an automatically allocated buffer. If *buf* is the NULL pointer, input/output will be completely unbuffered.

setvbuf can be used after a stream has been opened but before it is read or written. *type* determines how stream will be buffered. Legal values for *type* (defined in <stdio.h>) are:

_IOFBF fully buffers the input/output.

_IOLBF line buffers the output; the buffer will be flushed when a NEWLINE is written, the buffer is full, or input is requested.

_IONBF completely unbuffers the input/output.

If *buf* is not the NULL pointer, the array it points to will be used for buffering, instead of an automatically allocated buffer. *size* specifies the size of the buffer to be used.

setlinebuf is used to change the buffering on a stream from block buffered or unbuffered to line buffered. Unlike setbuf, setbuffer, and setvbuf, it can be used at any time that the file descriptor is active.

A file can be changed from unbuffered or line buffered to block buffered by using freopen (see fopen(3S)). A file can be changed from block buffered or line buffered to unbuffered by using freopen followed by setbuf with a buffer argument of NULL.

NOTE

A common source of error is allocating buffer space as an "automatic" variable in a code block, and then failing to close the stream in the same block.

SEE ALSO

fclose(3S), fopen(3S), fread(3S), getc(3S), malloc(3C), printf(3S), putc(3S), puts(3S), setbuf(3S) in the *Programmer's Reference Manual.*

RETURN VALUE

If an illegal value for *type* or *size* is provided, setvbuf returns a non-zero value. Otherwise, the value returned will be zero.

NAME

setbuffer, setlinebuf – assign buffering to a stream

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <stdio.h>

setbuffer(stream, buf, size)
FILE *stream;
char *buf;
int size;

setlinebuf(stream)
FILE *stream;
```

DESCRIPTION

The three types of buffering available are unbuffered, block buffered, and line buffered. When an output stream is unbuffered, information appears on the destination file or terminal as soon as written; when it is block buffered many characters are saved up and written as a block; when it is line buffered characters are saved up until a NEWLINE is encountered or input is read from any line buffered input stream. `fflush` (see `fclose`(3S)) may be used to force the block out early. Normally all files are block buffered. A buffer is obtained from `malloc`(3C) upon the first `getc` or `putc`(3S) on the file.

By default, output to a terminal is line buffered, except for output to the standard stream `stderr` which is unbuffered, and all other input/output is fully buffered.

`setbuffer` can be used after a stream has been opened but before it is read or written. It uses the character array *buf* whose size is determined by the *size* argument instead of an automatically allocated buffer. If *buf* is the `NULL` pointer, input/output will be completely unbuffered. A manifest constant `BUFSIZ`, defined in the `<stdio.h>` header file, tells how big an array is needed:

```
char buf[BUFSIZ];
```

`setlinebuf` is used to change the buffering on a stream from block buffered or unbuffered to line buffered. Unlike `setbuffer`, it can be used at any time that the file descriptor is active.

A file can be changed from unbuffered or line buffered to block buffered by using `freopen` (see `fopen`(3S)). A file can be changed from block buffered or line buffered to unbuffered by using `freopen` followed by `setbuffer` with a buffer argument of `NULL`.

SEE ALSO

`setbuf`(3S)

`fclose`(3S), `fopen`(3S), `fread`(3S), `getc`(3S), `malloc`(3C), `printf`(3S), `putc`(3S), `puts`(3S), `setbuf`(3S) in the *Programmer's Reference Manual.*

NOTE

A common source of error is allocating buffer space as an automatic variable in a code block, and then failing to close the stream in the same block.

NAME

setjmp, longjmp, _setjmp, _longjmp, sigsetjmp, siglongjmp – non-local goto

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
#include <setjmp.h>

int setjmp(env)
jmp_buf env;

longjmp(env, val)
jmp_buf env;
int val;

int _setjmp(env)
jmp_buf env;

_longjmp(env, val)
jmp_buf env;
int val;

int sigsetjmp(env, savemask)
sigjmp_buf env;
int savemask;

siglongjmp(env, val)
sigjmp_buf env;
int val;
```

DESCRIPTION

setjmp and longjmp are useful for dealing with errors and interrupts encountered in a low-level subroutine of a program.

setjmp saves its stack environment in *env* for later use by longjmp. A normal call to setjmp returns zero. setjmp also saves the register environment. If a longjmp call will be made, the routine which called setjmp should not return until after the longjmp has returned control (see below).

longjmp restores the environment saved by the last call of setjmp, and then returns in such a way that execution continues as if the call of setjmp had just returned the value *val* to the function that invoked setjmp; however, if *val* were zero, execution would continue as if the call of setjmp had returned one. This ensures that a "return" from setjmp caused by a call to longjmp can be distinguished from a regular return from setjmp. The calling function must not itself have returned in the interim, otherwise longjmp will be returning control to a possibly non-existent environment. All memory-bound data have values as of the time longjmp was called. The CPU and floating-point data registers are restored to the values they had at the time that setjmp was called. But, because the register storage class is only a hint to the C compiler, variables declared as register variables may not necessarily be assigned to machine registers, so their values are unpredictable after a longjmp. This is especially a problem for programmers trying to write machine-independent C routines.

setjmp and longjmp save and restore the signal mask (see sigsetmask(2)), while _setjmp and _longjmp manipulate only the C stack and registers. If the *savemask* flag to sigsetjmp is non-zero, the signal mask is saved, and a subsequent siglongjmp using the same *env* will restore the signal mask. If the *savemask* flag is zero, the signal mask is not saved, and a subsequent siglongjmp using the same *env* will not restore the signal mask. In all other ways, _setjmp and sigsetjmp function in the same way that setjmp does, and _longjmp and siglongjmp function in the same way that longjmp does.

None of these functions save or restore any floating-point status or control registers.

EXAMPLE

The following code fragment indicates the flow of control of the setjmp and longjmp combination:

```
function declaration
...
        jmp_buf    my_environment;
        ...
        if (setjmp(my_environment)) {
          /* register variables have unpredictable values */
              code after the return from longjmp
              ...
        } else {
          /* do not modify register vars in this leg of code */
              this is the return from setjmp
              ...
        }
```

SEE ALSO

cc(1), signal(3), sigsetmask(3), sigvec(3).

cc(1), signal(2), setjmp(3C) in the *Programmer's Reference Manual.*

BUGS

setjmp does not save the current notion of whether the process is executing on the signal stack. The result is that a longjmp to some place on the signal stack leaves the signal stack state incorrect.

On some systems setjmp also saves the register environment. Therefore, all data that are bound to registers are restored to the values they had at the time that setjmp was called. All memory-bound data have values as of the time longjmp was called. However, because the register storage class is only a hint to the C compiler, variables declared as register variables may not necessarily be assigned to machine registers, so their values are unpredictable after a longjmp. When using compiler options that specify automatic register allocation (see cc(1V)), the compiler will not attempt to assign variables to registers in routines that call setjmp.

longjmp never causes setjmp to return zero, so programmers should not depend on longjmp being able to cause setjmp to return zero.

NAME

setregid – set real and effective group IDs

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
int setregid(rgid, egid)
int rgid, egid;
```

DESCRIPTION

setregid is used to set the real and effective group IDs of the calling process. If *rgid* is −1, the real GID is not changed; if *egid* is −1, the effective GID is not changed. The real and effective GIDs may be set to different values in the same call.

If the effective user ID of the calling process is super-user, the real GID and the effective GID can be set to any legal value.

If the effective user ID of the calling process is not super-user, either the real GID can be set to the saved setGID from execv, or the effective GID can either be set to the saved setGID or the real GID. Note: if a setGID process sets its effective GID to its real GID, it can still set its effective GID back to the saved setGID.

In either case, if the real GID is being changed (that is, if *rgid* is not −1), or the effective GID is being changed to a value not equal to the real GID, the saved set-GID is set equal to the new effective GID.

If the real GID is changed from its current value, the old value is removed from the groups access list (see getgroups(2)) if it is present in that list, and the new value is added to the groups access list if it is not already present and if this would not cause the number of groups in that list to exceed NGROUPS, as defined in /usr/include/sys/param.h.

RETURN VALUE

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and errno is set to indicate the error.

ERRORS

setregid will fail and neither of the group IDs will be changed if:

EPERM — The calling process's effective UID is not the super-user and a change other than changing the real GID to the saved setGID, or changing the effective GID to the real GID or the saved GID, was specified.

SEE ALSO

setreuid(3)

exec(2), getuid(2), setuid(2) in the *Programmer's Reference Manual.*

NAME

setreuid – set real and effective user IDs

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
int setreuid(ruid, euid)
int ruid, euid;
```

DESCRIPTION

setreuid is used to set the real and effective user IDs of the calling process. If *ruid* is −1, the real user ID is not changed; if *euid* is −1, the effective user ID is not changed. The real and effective user IDs may be set to different values in the same call.

If the effective user ID of the calling process is super-user, the real user ID and the effective user ID can be set to any legal value.

If the effective user ID of the calling process is not super-user, either the real user ID can be set to the effective user ID, or the effective user ID can either be set to the saved set-user ID from execv or the real user ID. Note: if a set-UID process sets its effective user ID to its real user ID, it can still set its effective user ID back to the saved set-user ID.

In either case, if the real user ID is being changed (that is, if *ruid* is not −1), or the effective user ID is being changed to a value not equal to the real user ID, the saved set-user ID is set equal to the new effective user ID.

RETURN VALUE

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and errno is set to indicate the error.

ERRORS

setreuid will fail and neither of the user IDs will be changed if:

EPERM	The calling process's effective user ID is not the super-user and a change other than changing the real user ID to the effective user ID, or changing the effective user ID to the real user ID or the saved set-user ID, was specified.

SEE ALSO

setregid(3)

exec(2), getuid(2), setuid(2) in the *Programmer's Reference Manual.*

NAME

sigblock, sigmask – block signals

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <signal.h>

sigblock(mask);
int mask;

#define sigmask(signum)
```

DESCRIPTION

sigblock adds the signals specified in *mask* to the set of signals currently being blocked from delivery. Signals are blocked if the appropriate bit in *mask* is a 1; the macro sigmask is provided to construct the mask for a given *signum*. The previous mask is returned, and may be restored using sigsetmask(3).

It is not possible to block SIGKILL, SIGSTOP, or SIGCONT; this restriction is silently imposed by the system.

RETURN VALUE

The previous set of masked signals is returned.

SEE ALSO

sigsetmask(2), sigvec(2)

kill(2), sigaction(2), signal(2) in the *Programmer's Reference Manual.*

NAME

sigfpe – signal handling for specific SIGFPE codes

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <signal.h>

#include <floatingpoint.h>

sigfpe_handler_type sigfpe(code, hdl)
sigfpe_code_type code;
sigfpe_handler_type hdl;
```

DESCRIPTION

This function allows signal handling to be specified for particular SIGFPE codes. A call to sigfpe defines a new handler *hdl* for a particular SIGFPE *code* and returns the old handler as the value of the function sigfpe. Normally handlers are specified as pointers to functions; the special cases SIGFPE_IGNORE, SIGFPE_ABORT, and SIGFPE_DEFAULT allow ignoring, specifying core dump using abort(3), or default handling respectively.

For these IEEE-related codes:

FPE_FLTINEX_TRAP	fp_inexact	floating inexact result
FPE_FLTDIV_TRAP	fp_division	floating division by zero
FPE_FLTUND_TRAP	fp_underflow	floating underflow
FPE_FLTOVF_TRAP	fp_overflow	floating overflow
FPE_FLTBSUN_TRAP	fp_invalid	branch or set on unordered
FPE_FLTOPERR_TRAP	fp_invalid	floating operand error
FPE_FLTNAN_TRAP	fp_invalid	floating Not-A-Number

default handling is defined to be to call the handler specified to ieee_handler(3M).

For all other SIGFPE codes, default handling is to core dump using abort(3).

The compilation option -ffpa causes fpa recomputation to replace the default abort action for code FPE_FPA_ERROR. Note: SIGFPE_DEFAULT will restore abort rather than FPA recomputation for this code.

Three steps are required to intercept an IEEE-related SIGFPE code with sigfpe:

1) Set up a handler with sigfpe.
2) Enable the relevant IEEE trapping capability in the hardware, perhaps by using assembly-language instructions.
3) Perform a floating-point operation that generates the intended IEEE exception.

Unlike ieee_handler(3M), sigfpe never changes floating-point hardware mode bits affecting IEEE trapping. No IEEE-related SIGFPE signals will be generated unless those hardware mode bits are enabled.

SIGFPE signals can be handled using sigvec(2), signal(3), sigfpe(3), or ieee_handler(3M). In a particular program, to avoid confusion, use only one of these interfaces to handle SIGFPE signals.

EXAMPLE

A user-specified signal handler might look like this:

```
void sample_handler( sig, code, scp, addr )
        int sig ;              /* sig == SIGFPE always */
        int code ;
        struct sigcontext *scp ;
        char *addr ;
        {
                /*
                   Sample user-written sigfpe code handler.
                   Prints a message and continues.
                   struct sigcontext is defined in <signal.h>.
                 */
                printf(" ieee exception code %x occurred at pc %X \n",
                        code,scp->sc_pc);
        }
```

and it might be set up like this:

```
        extern void sample_handler;
        main
        {
                sigfpe_handler_type hdl, old_handler1, old_handler2;
        /*
         * save current overflow and invalid handlers; set the new
         * overflow handler to sample_handler and set the new
         * invalid handler to SIGFPE_ABORT (abort on invalid)
         */
                hdl = (sigfpe_handler_type) sample_handler;
                old_handler1 = sigfpe(FPE_FLTOVF_TRAP, hdl);
                old_handler2 = sigfpe(FPE_FLTOPERR_TRAP, SIGFPE_ABORT);
                ...
        /*
         * restore old overflow and invalid handlers
         */
                sigfpe(FPE_FLTOVF_TRAP,   old_handler1);
                sigfpe(FPE_FLTOPERR_TRAP, old_handler2);
        }
```

FILES

/usr/include/floatingpoint.h
/usr/include/signal.h

SEE ALSO

sigvec(2), floatingpoint(3), ieee_handler(3M), signal(3), abort(3C) in the *Programmer's Reference Manual.*

RETURN VALUE

sigfpe returns BADSIG if *code* is not zero or a defined SIGFPE code.

NAME

siginterrupt – allow signals to interrupt system calls

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
int siginterrupt(sig, flag)
int sig, flag;
```

DESCRIPTION

siginterrupt is used to change the system call restart behavior when a system call is interrupted by the specified signal. If the flag is false (0), then system calls will be restarted if they are interrupted by the specified signal and no data has been transferred yet. System call restart is the default behavior when the signal(3) routine is used.

If the flag is true (1), then restarting of system calls is disabled. If a system call is interrupted by the specified signal and no data has been transferred, the system call will return −1 with errno set to EINTR. Interrupted system calls that have started transferring data will return the amount of data actually transferred.

Issuing a siginterrupt call during the execution of a signal handler will cause the new action to take place on the next signal to be caught.

NOTES

This library routine uses an extension of the sigvec(2) system call that is not available in 4.2BSD, hence it should not be used if backward compatibility is needed.

RETURN VALUE

A 0 value indicates that the call succeeded. A −1 value indicates that an invalid signal number has been supplied.

SEE ALSO

sigblock(3), sigpause(3), sigsetmask(3), sigvec(3), signal(3).

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

NAME

signal – simplified software signal facilities

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
#include <signal.h>

void (*signal(sig, func))()
void (*func)();
```

DESCRIPTION

signal is a simplified interface to the more general sigvec(2) facility. Programs that use signal in preference to sigvec are more likely to be portable to all systems.

A signal is generated by some abnormal event, initiated by a user at a terminal (quit, interrupt, stop), by a program error (bus error, etc.), by request of another program (kill), or when a process is stopped because it wishes to access its control terminal while in the background [see termio(4)]. Signals are optionally generated when a process resumes after being stopped, when the status of child processes changes, or when input is ready at the control terminal. Most signals cause termination of the receiving process if no action is taken; some signals instead cause the process receiving them to be stopped, or are simply discarded if the process has not requested otherwise. Except for the SIGKILL and SIGSTOP signals, the signal call allows signals either to be ignored or to interrupt to a specified location. The following is a list of all signals with names as in the include file <signal.h>:

SIGHUP		hangup
SIGINT		interrupt
SIGQUIT	*	quit
SIGILL	*	illegal instruction
SIGTRAP	*	trace trap
SIGABRT	*	abort (generated by abort(3) routine)
SIGEMT	*	emulator trap
SIGFPE	*	arithmetic exception
SIGKILL		kill (cannot be caught, blocked, or ignored)
SIGBUS	*	bus error
SIGSEGV	*	segmentation violation
SIGSYS	*	bad argument to system call
SIGPIPE		write on a pipe or other socket with no one to read it
SIGALRM		alarm clock
SIGTERM		software termination signal
SIGURG	•	urgent condition present on socket
SIGSTOP	†	stop (cannot be caught, blocked, or ignored)
SIGTSTP	†	stop signal generated from keyboard
SIGCONT	•	continue after stop (cannot be blocked)
SIGCHLD	•	child status has changed
SIGTTIN	†	background read attempted from control terminal
SIGTTOU	†	background write attempted to control terminal
SIGIO	•	I/O is possible on a descriptor [see fcntl(2)]
SIGXCPU	*	cpu time limit exceeded [see getrlimit(2)

SIGXFSZ	*	file size limit exceeded [see getrlimit(2)
SIGVTALRM		virtual time alarm [see getitimer(2)
SIGPROF		profiling timer alarm [see getitimer(2)]
SIGWINCH	•	window changed [see termio(4)]
SIGUSR1		user-defined signal 1
SIGUSR2		user-defined signal 2

The starred signals in the list above cause a core image if not caught or ignored.

If *func* is SIG_DFL, the default action for signal *sig* is reinstated; this default is termination (with a core image for starred signals) except for signals marked with • or †. Signals marked with • are discarded if the action is SIG_DFL; signals marked with † cause the process to stop. If *func* is SIG_IGN the signal is subsequently ignored and pending instances of the signal are discarded. Otherwise, when the signal occurs further occurrences of the signal are automatically blocked and *func* is called.

A return from the function unblocks the handled signal and continues the process at the point it was interrupted.

If a caught signal occurs during certain system calls, terminating the call prematurely, the call is automatically restarted. In particular this can occur during a read(2) or write(2) on a slow device (such as a terminal; but not a file) and during a wait(2).

The value of signal is the previous (or initial) value of *func* for the particular signal.

After a fork(2) or vfork(2) the child inherits all signals. An execve(2) resets all caught signals to the default action; ignored signals remain ignored.

NOTES

The handler routine can be declared:

```
void handler(sig, code, scp, addr)
int sig, code;
struct sigcontext *scp;
char *addr;
```

Here *sig* is the signal number; *code* is a parameter of certain signals that provides additional detail; *scp* is a pointer to the sigcontext structure (defined in <signal.h>), used to restore the context from before the signal; and *addr* is additional address information. See sigvec(2) for more details.

RETURN VALUE

The previous action is returned on a successful call. Otherwise, −1 is returned and errno is set to indicate the error.

ERRORS

signal will fail and no action will take place if one of the following occur:

EINVAL	*sig* is not a valid signal number, or is SIGKILL or SIGSTOP.

SEE ALSO

setjmp(3), sigblock(3), sigpause(3), sigsetmask(3), sigstack(3), sigvec(3), wait(3)

execve(2), fork(2), getitimer(2), getrlimit(2), kill(2), ptrace(2), read(2), sigaction(2) wait(2), write(2), setjmp(3C), in the *Programmer's Reference Manual.*
kill(1), in the *User's Reference Manual.*
termio(7) in the *System Administrator's Reference Manual.*

NAME

sigpause – automically release blocked signals and wait for interrupt

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
sigpause(sigmask)
int sigmask;
```

DESCRIPTION

sigpause assigns *sigmask* to the set of masked signals and then waits for a signal to arrive; on return the set of masked signals is restored. *sigmask* is usually 0 to indicate that no signals are now to be blocked. sigpause always terminates by being interrupted, returning EINTR.

In normal usage, a signal is blocked using sigblock(3), to begin a critical section, variables modified on the occurrence of the signal are examined to determine that there is no work to be done, and the process pauses awaiting work by using sigpause with the mask returned by sigblock.

SEE ALSO

sigblock(3), sigvec(3), signal(3)

signal(2), sigaction(2) in the *Programmer's Reference Manual.*

NAME

sigsetmask – set current signal mask

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
#include <signal.h>

sigsetmask(mask);
int mask;

#define sigmask(signum)
```

DESCRIPTION

sigsetmask sets the current signal mask (those signals that are blocked from delivery). Signals are blocked if the corresponding bit in *mask* is a 1; the macro sigmask is provided to construct the mask for a given *signum*.

The system quietly disallows SIGKILL, SIGSTOP, or SIGCONT from being blocked.

RETURN VALUE

The previous set of masked signals is returned.

SEE ALSO

sigblock(3), sigpause(3), sigvec(3), signal(3)

kill(2), signal(2) in the *Programmer's Reference Manual.*

NAME

sigstack – set and/or get signal stack context

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <signal.h>

int sigstack (ss, oss)
struct sigstack *ss, *oss;
```

DESCRIPTION

sigstack allows users to define an alternate stack, called the "signal stack", on which signals are to be processed. When a signal's action indicates its handler should execute on the signal stack (specified with a sigvec(2) call), the system checks to see if the process is currently executing on that stack. If the process is not currently executing on the signal stack, the system arranges a switch to the signal stack for the duration of the signal handler's execution.

A signal stack is specified by a sigstack structure, which includes the following members:

```
char        *ss_sp;          /* signal stack pointer */
int         ss_onstack;      /* current status */
```

ss_sp is the initial value to be assigned to the stack pointer when the system switches the process to the signal stack. Note that, on machines where the stack grows downwards in memory, this is *not* the address of the beginning of the signal stack area. ss_onstack field is zero or non-zero depending on whether the process is currently executing on the signal stack or not.

If *ss* is not a NULL pointer, sigstack sets the signal stack state to the value in the sigstack structure pointed to by *ss*. Note: if ss_onstack is non-zero, the system will think that the process is executing on the signal stack. If *ss* is a NULL pointer, the signal stack state will be unchanged. If *oss* is not a NULL pointer, the current signal stack state is stored in the sigstack structure pointed to by *oss*.

RETURN VALUE

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and errno is set to indicate the error.

ERRORS

sigstack will fail and the signal stack context will remain unchanged if one of the following occurs.

EFAULT — Either *ss* or *oss* points to memory that is not a valid part of the process address space.

SEE ALSO

sigvec(3), signal(3)

sigaltstack(2), in the *Programmer's Reference Manual.*

NOTES

Signal stacks are not "grown" automatically, as is done for the normal stack. If the stack overflows unpredictable results may occur.

NAME

sigvec – software signal facilities

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <signal.h>

int sigvec(sig, vec, ovec)
int sig;
struct sigvec *vec, *ovec;
```

DESCRIPTION

The system defines a set of signals that may be delivered to a process. Signal delivery resembles the occurrence of a hardware interrupt: the signal is blocked from further occurrence, the current process context is saved, and a new one is built. A process may specify a *handler* to which a signal is delivered, or specify that a signal is to be *blocked* or *ignored*. A process may also specify that a default action is to be taken by the system when a signal occurs. Normally, signal handlers execute on the current stack of the process. This may be changed, on a per-handler basis, so that signals are taken on a special *signal stack*.

All signals have the same *priority*. Signal routines execute with the signal that caused their invocation to be *blocked*, but other signals may yet occur. A global *signal mask* defines the set of signals currently blocked from delivery to a process. The signal mask for a process is initialized from that of its parent (normally 0). It may be changed with a `sigblock`(3) or `sigsetmask`(3) call, or when a signal is delivered to the process.

A process may also specify a set of *flags* for a signal that affect the delivery of that signal.

When a signal condition arises for a process, the signal is added to a set of signals pending for the process. If the signal is not currently *blocked* by the process then it is delivered to the process. When a signal is delivered, the current state of the process is saved, a new signal mask is calculated (as described below), and the signal handler is invoked. The call to the handler is arranged so that if the signal handling routine returns normally the process will resume execution in the context from before the signal's delivery. If the process wishes to resume in a different context, then it must arrange to restore the previous context itself.

When a signal is delivered to a process a new signal mask is installed for the duration of the process' signal handler (or until a `sigblock` or `sigsetmask` call is made). This mask is formed by taking the current signal mask, adding the signal to be delivered, and ORing in the signal mask associated with the handler to be invoked.

The action to be taken when the signal is delivered is specified by a `sigvec` structure, which includes the following members:

```
void    (*sv_handler)();        /* signal handler */
int     sv_mask;                /* signal mask to apply */
int     sv_flags;               /* see signal options */
```

```
#define SV_ONSTACK    /* take signal on signal stack */
#define SV_INTERRUPT  /* do not restart system on signal return */
#define SV_RESETHAND  /* reset handler to SIG_DFL when signal taken */
```

If the SV_ONSTACK bit is set in the flags for that signal, the system will deliver the signal to the process on the signal stack specified with sigstack(2), rather than delivering the signal on the current stack.

If *vec* is not a NULL pointer, sigvec assigns the handler specified by sv_handler, the mask specified by sv_mask, and the flags specified by sv_flags to the specified signal. If *vec* is a NULL pointer, sigvec does not change the handler, mask, or flags for the specified signal.

The mask specified in *vec* is not allowed to block SIGKILL, SIGSTOP, or SIGCONT. The system enforces this restriction silently.

If *ovec* is not a NULL pointer, the handler, mask, and flags in effect for the signal before the call to sigvec are returned to the user. A call to sigvec with *vec* a NULL pointer and *ovec* not a NULL pointer can be used to determine the handling information currently in effect for a signal without changing that information.

The following is a list of all signals with names as in the include file /usr/include/signal.h:

SIGHUP		hangup
SIGINT		interrupt
SIGQUIT	*	quit
SIGILL	*	illegal instruction
SIGTRAP	*	trace trap
SIGABRT	*	abort (generated by abort(3) routine)
SIGEMT	*	emulator trap
SIGFPE	*	arithmetic exception
SIGKILL		kill (cannot be caught, blocked, or ignored)
SIGBUS	*	bus error
SIGSEGV	*	segmentation violation
SIGSYS	*	bad argument to system call
SIGPIPE		write on a pipe or other socket with no one to read it
SIGALRM		alarm clock
SIGTERM		software termination signal
SIGURG	•	urgent condition present on socket
SIGSTOP	†	stop (cannot be caught, blocked, or ignored)
SIGTSTP	†	stop signal generated from keyboard
SIGCONT	•	continue after stop (cannot be blocked)
SIGCHLD	•	child status has changed
SIGTTIN	†	background read attempted from control terminal
SIGTTOU	†	background write attempted to control terminal
SIGIO	•	I/O is possible on a descriptor [see fcntl(2)]
SIGXCPU		cpu time limit exceeded [see setrlimit(2)]
SIGXFSZ		file size limit exceeded [see setrlimit(2)]
SIGVTALRM		virtual time alarm [see setitimer(2)]
SIGPROF		profiling timer alarm [see setitimer(2)]
SIGWINCH	•	window changed [see termio(4)]
SIGUSR1		user-defined signal 1

SIGUSR2 user-defined signal 2

The starred signals in the list above cause a core image if not caught or ignored.

Once a signal handler is installed, it remains installed until another sigvec call is made, or an execve(2) is performed, unless the SV_RESETHAND bit is set in the flags for that signal. In that case, the value of the handler for the caught signal will be set to SIG_DFL before entering the signal-catching function, unless the signal is SIGILL, SIGPWR, or SIGTRAP. Also, if this bit is set, the bit for that signal in the signal mask will not be set; unless the signal mask associated with that signal blocks that signal, further occurrences of that signal will not be blocked. The SV_RESETHAND flag is not available in 4.2BSD, hence it should not be used if backward compatibility is needed.

The default action for a signal may be reinstated by setting the signal's handler to SIG_DFL; this default is termination except for signals marked with • or †. Signals marked with • are discarded if the action is SIG_DFL; signals marked with † cause the process to stop. If the process is terminated, a "core image" will be made in the current working directory of the receiving process if the signal is one for which an asterisk appears in the above list [see core(4)].

If the handler for that signal is SIG_IGN, the signal is subsequently ignored, and pending instances of the signal are discarded.

If a caught signal occurs during certain system calls, the call is normally restarted. The call can be forced to terminate prematurely with an EINTR error return by setting the SV_INTERRUPT bit in the flags for that signal. The SV_INTERRUPT flag is not available in 4.2BSD, hence it should not be used if backward compatibility is needed. The affected system calls are read(2) or write(2) on a slow device (such as a terminal or pipe or other socket, but not a file) and during a wait(2).

After a fork(2) or vfork(2) the child inherits all signals, the signal mask, the signal stack, and the restart/interrupt and reset-signal-handler flags.

The execve(2) call resets all caught signals to default action and resets all signals to be caught on the user stack. Ignored signals remain ignored; the signal mask remains the same; signals that interrupt system calls continue to do so.

The accuracy of *addr* is machine dependent. For example, certain machines may supply an address that is on the same page as the address that caused the fault. If an appropriate *addr* cannot be computed it will be set to SIG_NOADDR.

RETURN VALUE

A 0 value indicates that the call succeeded. A −1 return value indicates that an error occurred and errno is set to indicate the reason.

ERRORS

sigvec will fail and no new signal handler will be installed if one of the following occurs:

EFAULT Either *vec* or *ovec* is not a NULL pointer and points to memory that is not a valid part of the process address space.

EINVAL *Sig* is not a valid signal number, or, SIGKILL, or SIGSTOP.

SEE ALSO

signal(3), sigpause(3), sigsetmask(3), wait(3).

exec(2), fcntl(2), fork(2), getrlimit(2), getitimer(2), ioctl(2), kill(2), ptrace(2), read(2), sigblock(2), signal(2), sigstack(2), umask(2), wait(2), write(2), setjmp(3) in the *Programmer's Reference Manual.*
streamio(7), termio(7) in the *System Administrator's Reference Manual.*

NOTES

SIGPOLL is a synonym for SIGIO. A SIGIO will be issued when a file descriptor corresponding to a STREAMS [see intro(2)] file has a "selectable" event pending. Unless that descriptor has been put into asynchronous mode [see fcntl(2)], a process must specifically request that this signal be sent using the I_SETSIG ioctl call [see streamio(4)]. Otherwise, the process will never receive SIGPOLL.

The handler routine can be declared:

```
void handler(sig, code, scp, addr)
int sig, code;
struct sigcontext *scp;
char *addr;
```

Here *sig* is the signal number; *code* is a parameter of certain signals that provides additional detail; *scp* is a pointer to the sigcontext structure (defined in signal.h), used to restore the context from before the signal; and *addr* is additional address information.

The signals SIGKILL, SIGSTOP, and SIGCONT cannot be ignored.

NAME

sleep – suspend execution for interval

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
sleep(seconds)
unsigned seconds;
```

DESCRIPTION

sleep suspends the current process from execution for the number of seconds specified by the argument. The actual suspension time may be up to 1 second less than that requested, because scheduled wakeups occur at fixed 1-second intervals, and may be an arbitrary amount longer because of other activity in the system.

sleep is implemented by setting an interval timer and pausing until it expires. The previous state of this timer is saved and restored. If the sleep time exceeds the time to the expiration of the previous value of the timer, the process sleeps only until the timer would have expired, and the signal which occurs with the expiration of the timer is sent one second later.

SEE ALSO

sigpause(3), usleep(3)

getitimer(2) in the *Programmer's Reference Manual*

NAME

string: strcasecmp, strncasecmp – string operations

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
int strcasecmp(s1, s2)
char *s1, *s2;

int strncasecmp(s1, s2, n)
char *s1, *s2;
int n;
```

DESCRIPTION

The strcasecmp and strncasecmp routines compare the strings and ignore differences in case. These routines assume the ASCII character set when equating lower and upper case characters.

These functions operate on null-terminated strings. They do not check for overflow of any receiving string.

SEE ALSO

bstring(3)

malloc(3C), string(3C) in the *Programmer's Reference Manual.*

NOTES

strcasecmp and strncasecmp use native character comparison as above and assume the ASCII character set.

NAME

syscall – indirect system call

SYNOPSIS

cc [*flag...*] *file* ... -lucb

```
#include <sys/syscall.h>
```

```
int syscall(number, arg, ...)
```

DESCRIPTION

syscall performs the system call whose assembly language interface has the specified *number*, and arguments *arg* .... Symbolic constants for system calls can be found in the header file /usr/include/sys/syscall.h.

RETURN VALUE

When the C-bit is set, syscall returns −1 and sets the external variable errno (see intro(2)).

SEE ALSO

intro(2), pipe(2) in the *Programmer's Reference Manual.*

NAME

syslog, openlog, closelog, setlogmask – control system log

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <syslog.h>

openlog(ident, logopt, facility)
char *ident;

syslog(priority, message, parameters . . . )
char *message;

closelog()

setlogmask(maskpri)
```

DESCRIPTION

syslog passes *message* to syslogd(1M), which logs it in an appropriate system log, writes it to the system console, forwards it to a list of users, or forwards it to the syslogd on another host over the network. The message is tagged with a priority of *priority*. The message looks like a printf(3S) string except that %m is replaced by the current error message (collected from errno). A trailing NEWLINE is added if needed.

Priorities are encoded as a *facility* and a *level*. The facility describes the part of the system generating the message. The level is selected from an ordered list:

LOG_EMERG	A panic condition. This is normally broadcast to all users.
LOG_ALERT	A condition that should be corrected immediately, such as a corrupted system database.
LOG_CRIT	Critical conditions, such as hard device errors.
LOG_ERR	Errors.
LOG_WARNING	Warning messages.
LOG_NOTICE	Conditions that are not error conditions, but that may require special handling.
LOG_INFO	Informational messages.
LOG_DEBUG	Messages that contain information normally of use only when debugging a program.

If special processing is needed, openlog can be called to initialize the log file. The parameter *ident* is a string that is prepended to every message. *logopt* is a bit field indicating logging options. Current values for *logopt* are:

LOG_PID	Log the process ID with each message. This is useful for identifying specific daemon processes (for daemons that fork).
LOG_CONS	Write messages to the system console if they cannot be sent to syslogd. This option is safe to use in daemon processes that have no controlling terminal, since syslog forks before opening the console.

LOG_NDELAY	Open the connection to syslogd immediately. Normally the open is delayed until the first message is logged. This is useful for programs that need to manage the order in which file descriptors are allocated.
LOG_NOWAIT	Do not wait for child processes that have been forked to log messages onto the console. This option should be used by processes that enable notification of child termination using SIGCHLD, since syslog may otherwise block waiting for a child whose exit status has already been collected.

The *facility* parameter encodes a default facility to be assigned to all messages that do not have an explicit facility already encoded:

LOG_KERN	Messages generated by the kernel. These cannot be generated by any user processes.
LOG_USER	Messages generated by random user processes. This is the default facility identifier if none is specified.
LOG_MAIL	The mail system.
LOG_DAEMON	System daemons, such as ftpd(1M), routed(1M), etc.
LOG_AUTH	The authorization system: login(1), su(1), getty(1M), etc.
LOG_LPR	The line printer spooling system: lpr(1), lpc(1M), etc.
LOG_NEWS	Reserved for the USENET network news system.
LOG_UUCP	Reserved for the UUCP system; it does not currently use syslog.
LOG_CRON	The cron/at facility; crontab(1), at(1), cron(1M), etc.
LOG_LOCAL0-7	Reserved for local use.

closelog can be used to close the log file.

setlogmask sets the log priority mask to *maskpri* and returns the previous mask. Calls to syslog with a priority not set in *maskpri* are rejected. The mask for an individual priority *pri* is calculated by the macro LOG_MASK(*pri*); the mask for all priorities up to and including *toppri* is given by the macro LOG_UPTO(*toppri*). The default allows all priorities to be logged.

EXAMPLE

This call logs a message at priority LOG_ALERT:

```
syslog(LOG_ALERT, "who: internal error 23");
```

The FTP daemon, ftpd, would make this call to openlog to indicate that all messages it logs should have an identifying string of ftpd, should be treated by syslogd as other messages from system daemons are, and should include the process ID of the process logging the message:

```
openlog("ftpd", LOG_PID, LOG_DAEMON);
```

Then it would make the following call to setlogmask to indicate that messages at priorities from LOG_EMERG through LOG_ERR should be logged, but that no messages at any other priority should be logged:

```
setlogmask(LOG_UPTO(LOG_ERR));
```

Then, to log a message at priority LOG_INFO, it would make the following call to syslog:

```
syslog(LOG_INFO, "Connection from host %d", CallingHost);
```

A locally-written utility could use the following call to syslog to log a message at priority LOG_INFO, to be treated by syslogd as other messages to the facility LOG_LOCAL2 are treated:

```
syslog(LOG_INFO|LOG_LOCAL2, "error: %m");
```

SEE ALSO

logger(1), login(1), lpr(1), lpc(1M), syslogd(1M), printf(3S)

at(1), crontab(1), login(1) in the *User's Reference Manual.*
ftpd(1M), routed(1M) in the *Network User's and Administrator's Guide.*
getty(1M), cron(1M), su(1), printf(3S) in the *System Administrator's Reference Manual.*

NAME

times – get process times

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <sys/types.h>
#include <sys/times.h>

times(buffer)
struct tms *buffer;
```

DESCRIPTION

times returns time-accounting information for the current process and for the terminated child processes of the current process. All times are in 1/HZ seconds, where HZ is 60.

This is the structure returned by times:

```
struct tms {
        time_t tms_utime;     /* user time */
        time_t tms_stime;     /* system time */
        time_t tms_cutime;    /* user time, children */
        time_t tms_cstime;    /* system time, children */
};
```

The children's times are the sum of the children's process times and their children's times.

SEE ALSO

getrusage(3), wait(3), time(3)

time(1) in the *User's Reference Manual.*
wait(2) in the *Programmer's Reference Manual*

NOTES

times has been superseded by getrusage.

NAME

timezone – get time zone name given offset from GMT

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
char *timezone(zone, dst)

int zone
int dst
```

DESCRIPTION

timezone attempts to return the name of the time zone associated with its first argument, which is measured in minutes westward from Greenwich. If the second argument is 0, the standard name is used, otherwise the Daylight Savings Time version. If the required name does not appear in a table built into the routine, the difference from GMT is produced; for instance, in Afghanistan timezone(-(60*4+30), 0) is appropriate because it is 4:30 ahead of GMT and the string GMT+4:30 is produced.

SEE ALSO

ctime(3) in the *Programmer's Reference Manual.*

NOTES

The offset westward from Greenwich and an indication of whether Daylight Savings Time is in effect may not be sufficient to determine the name of the time zone, as the name may differ between different locations in the same time zone. Instead of using timezone to determine the name of the time zone for a given time, that time should be converted to a struct tm using localtime [see ctime(3)] and the tm_zone field of that structure should be used. timezone is retained for compatibility with existing programs.

NAME

ualarm – schedule signal after interval in microseconds

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
unsigned ualarm(value, interval)

unsigned value;
unsigned interval;
```

DESCRIPTION

ualarm sends signal SIGALRM [see signal(3)], to the invoking process in a number of microseconds given by the *value* argument. Unless caught or ignored, the signal terminates the process.

If the *interval* argument is non-zero, the SIGALRM signal will be sent to the process every *interval* microseconds after the timer expires (for instance, after *value* microseconds have passed).

Because of scheduling delays, resumption of execution of when the signal is caught may be delayed an arbitrary amount. The longest specifiable delay time is 2147483647 microseconds.

The return value is the amount of time previously remaining in the alarm clock.

NOTES

ualarm is a simplified interface to setitimer; see getitimer(2).

SEE ALSO

sigpause(3), sigvec(3), signal(3), sleep(3), usleep(3)

alarm(2), getitimer(3) in the *Programmer's Reference Manual.*

NAME

usleep – suspend execution for interval in microseconds

SYNOPSIS

cc [*flag...*] *file ...* -lucb

```
usleep(useconds)
unsigned useconds;
```

DESCRIPTION

Suspend the current process for the number of microseconds specified by the argument. The actual suspension time may be an arbitrary amount longer because of other activity in the system, or because of the time spent in processing the call.

The routine is implemented by setting an interval timer and pausing until it occurs. The previous state of this timer is saved and restored. If the sleep time exceeds the time to the expiration of the previous timer, the process sleeps only until the signal would have occurred, and the signal is sent a short time later.

This routine is implemented using `setitimer` [see `getitimer`(2)]; it requires eight system calls each time it is invoked.

SEE ALSO

`sigpause`(3), `sleep`(3), `ualarm`(3).

`getitimer`(3), `alarm`(2) in the *Programmer's Reference Manual.*

NAME

utimes – set file times

SYNOPSIS

cc [*flag*...] *file* ... -lucb

```
#include <sys/types.h>

int utimes(file, tvp)
char *file;
struct timeval *tvp;
```

DESCRIPTION

utimes sets the access and modification times of the file named by *file*.

If *tvp* is NULL, the access and modification times are set to the current time. A process must be the owner of the file or have write permission for the file to use utimes in this manner.

If *tvp* is not NULL, it is assumed to point to an array of two timeval structures. The access time is set to the value of the first member, and the modification time is set to the value of the second member. Only the owner of the file or the privileged user may use utimes in this manner.

In either case, the *inode-changed* time of the file is set to the current time.

RETURN VALUE

Upon successful completion, a value of 0 is returned. Otherwise, a value of −1 is returned and errno is set to indicate the error.

ERRORS

utimes will fail if one or more of the following are true:

ENOTDIR	A component of the path prefix of *file* is not a directory.
ENAMETOOLONG	The length of a component of *file* exceeds 255 characters, or the length of *file* exceeds 1023 characters.
ENOENT	The file referred to by *file* does not exist.
EACCES	Search permission is denied for a component of the path prefix of *file*.
ELOOP	Too many symbolic links were encountered in translating *file*.
EPERM	The effective user ID of the process is not privileged user and not the owner of the file, and *tvp* is not NULL.
EACCES	The effective user ID of the process is not privileged user and not the owner of the file, write permission is denied for the file, and *tvp* is NULL.
EIO	An I/O error occurred while reading from or writing to the file system.
EROFS	The file system containing the file is mounted read-only.

EFAULT	*file* or *tvp* points outside the process's allocated address space.

SEE ALSO

stat(2), utime(2) in the *Programmer's Reference Manual.*

NOTES

utimes is a library routine that calls the utime system call.

NAME

wait, wait3, WIFSTOPPED, WIFSIGNALED, WIFEXITED – wait for process to terminate or stop

SYNOPSIS

```
cc [ flag... ] file ... -lucb

#include <sys/wait.h>

int wait(statusp)
union wait *statusp;

#include <sys/time.h>
#include <sys/resource.h>

int wait3(statusp, options, rusage)
union wait *statusp;
int options;
struct rusage *rusage;

WIFSTOPPED(status)
union wait status;

WIFSIGNALED(status)
union wait status;

WIFEXITED(status)
union wait status;
```

DESCRIPTION

wait delays its caller until a signal is received or one of its child processes terminates or stops due to tracing. If any child has died or stopped due to tracing and this has not been reported using wait, return is immediate, returning the process ID and exit status of one of those children. If that child had died, it is discarded. If there are no children, return is immediate with the value −1 returned. If there are only running or stopped but reported children, the calling process is blocked.

If *status* is not a NULL pointer, then on return from a successful wait call the status of the child process whose process ID is the return value of wait is stored in the wait union pointed to by *status*. The w_status member of that union is an int; it indicates the cause of termination and other information about the terminated process in the following manner:

- If the low-order 8 bits of w_status are equal to 0177, the child process has stopped; the 8 bits higher up from the low-order 8 bits of w_status contain the number of the signal that caused the process to stop. See ptrace(2) and sigvec(3).
- If the low-order 8 bits of w_status are non-zero and are not equal to 0177, the child process terminated due to a signal; the low-order 7 bits of w_status contain the number of the signal that terminated the process. In addition, if the low-order seventh bit of w_status (that is, bit 0200) is set, a "core image" of the process was produced; see sigvec(3).

- Otherwise, the child process terminated due to an `exit` call; the 8 bits higher up from the low-order 8 bits of `w_status` contain the low-order 8 bits of the argument that the child process passed to `exit`; see `exit(2)`.

Other members of the `wait` union can be used to extract this information more conveniently:

- If the `w_stopval` member has the value `WSTOPPED`, the child process has stopped; the value of the `w_stopsig` member is the signal that stopped the process.
- If the `w_termsig` member is non-zero, the child process terminated due to a signal; the value of the `w_termsig` member is the number of the signal that terminated the process. If the `w_coredump` member is non-zero, a core dump was produced.
- Otherwise, the child process terminated due to an `exit` call; the value of the `w_retcode` member is the low-order 8 bits of the argument that the child process passed to `exit`.

The other members of the `wait` union merely provide an alternate way of analyzing the status. The value stored in the `w_status` field is compatible with the values stored by other versions of the UNIX system, and an argument of type `int *` may be provided instead of an argument of type `union wait *` for compatibility with those versions.

`wait3` is an alternate interface that allows both non-blocking status collection and the collection of the status of children stopped by any means. The *status* parameter is defined as above. The *options* parameter is used to indicate the call should not block if there are no processes that have status to report (`WNOHANG`), and/or that children of the current process that are stopped due to a `SIGTTIN`, `SIGTTOU`, `SIGTSTP`, or `SIGSTOP` signal are eligible to have their status reported as well (`WUNTRACED`). A terminated child is discarded after it reports status, and a stopped process will not report its status more than once. If *rusage* is not a `NULL` pointer, a summary of the resources used by the terminated process and all its children is returned. Only the user time used and the system time used are currently available. They are returned in `rusage.ru_utime` and `rusage.ru_stime`, respectively.

When the WNOHANG option is specified and no processes have status to report, `wait3` returns 0. The `WNOHANG` and `WUNTRACED` options may be combined by ORing the two values.

`WIFSTOPPED`, `WIFSIGNALED`, `WIFEXITED`, are macros that take an argument *status*, of type 'union wait', as returned by `wait`, or `wait3`. `WIFSTOPPED` evaluates to true (1) when the process for which the `wait` call was made is stopped, or to false (0) otherwise. `WIFSIGNALED` evaluates to true when the process was terminated with a signal. `WIFEXITED` evaluates to true whe the process exited by using an `exit(2)` call.

RETURN VALUE

If `wait` returns due to a stopped or terminated child process, the process ID of the child is returned to the calling process. Otherwise, a value of −1 is returned and `errno` is set to indicate the error.

wait3 returns 0 if WNOHANG is specified and there are no stopped or exited children, and returns the process ID of the child process if it returns due to a stopped or terminated child process. Otherwise, wait3 returns a value of −1 and sets errno to indicate the error.

ERRORS

wait, or wait3 will fail and return immediately if one or more of the following are true:

ECHILD The calling process has no existing unwaited-for child processes.

EFAULT The *status* or *rusage* arguments point to an illegal address.

wait, and wait3 will terminate prematurely, return −1, and set errno to EINTR upon the arrival of a signal whose SV_INTERRUPT bit in its flags field is set [see sigvec(3) and siginterrupt(3)]. signal(3), in the System V compatibility library, sets this bit for any signal it catches.

SEE ALSO

sigvec(3), getrusage(3), siginterrupt(3), signal(3)

exit(2), ptrace(2), signal(2) wait(2), waitpid(2) in the *Programmer's Reference Manual.*

NOTES

If a parent process terminates without waiting on its children, the initialization process (process ID = 1) inherits the children.

wait, and wait3 are automatically restarted when a process receives a signal while awaiting termination of a child process, unless the SV_INTERRUPT bit is set in the flags for that signal.

WARNINGS

Calls to wait with an argument of 0 should be cast to type 'union wait *', as in:

```
wait((union wait *)0)
```

Otherwise lint will complain.

FILE FORMATS (4)

NAME

aliases, addresses, forward – addresses and aliases for sendmail

SYNOPSIS

```
/etc/aliases
/etc/aliases.dir
/etc/aliases.pag
~/.forward
```

DESCRIPTION

These files contain mail addresses or aliases, recognized by sendmail, for the local host:

/etc/passwd	Mail addresses (usernames) of local users.
/etc/aliases	Aliases for the local host, in ASCII format. This file can be edited to add, update, or delete local mail aliases.
/etc/aliases. { dir , pag}	The aliasing information from /etc/aliases, in binary, dbm format for use by sendmail. The program newaliases, which is invoked automatically by sendmail, maintains these files.
~/.forward	Addresses to which a user's mail is forwarded (see Automatic Forwarding, below).

In addition, the YP name services aliases map *mail.aliases* contains addresses and aliases available for use across the network.

Addresses

As distributed, sendmail supports the following types of addresses:

Local Usernames

username

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

Local Filenames

pathname

Messages addressed to the absolute *pathname* of a file are appended to that file.

Commands

| *command*

If the first character of the address is a vertical bar, (|), sendmail pipes the message to the standard input of the *command* the bar precedes.

DARPA-standard Addresses

username@domain

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

.COM Commerical organizations.

.EDU Educational organizations.

.GOV Government organizations.

.MIL Military organizations.

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

`js@jsmachine.Podunk-U.EDU`

if he uses the machine named `jsmachine` at Podunk University.

uucp Addresses

... [*host*!]*host*!*username*

These are sometimes mistakenly referred to as "Usenet" addresses. uucp provides links to numerous sites throughout the world for the remote copying of files.

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

Aliases

Local Aliases

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

aliasname:*address*[, *address*]

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

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

Special Aliases

An alias of the form:

`owner-` *aliasname* : *address*

directs error-messages resulting from mail to *aliasname* to *address*, instead of back to the person who sent the message.

An alias of the form:

aliasname: `:include:`*pathname*

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

YP Domain Aliases

Normally, the aliases file on the master YP server is used for the *mail.aliases* YP map, which can be made available to every YP client. Thus, the `/etc/aliases*` files on the various hosts in a network will one day be obsolete. Domain-wide aliases should ultimately be resolved into usernames on specific hosts. For

example, if the following were in the domain-wide alias file:

```
jsmith:js@jsmachine
```

then any YP client could just mail to `jsmith` and not have to remember the machine and username for John Smith. If a YP alias does not resolve to an address with a specific host, then the name of the YP domain is used. There should be an alias of the domain name for a host in this case. For example, the alias:

```
jsmith:root
```

sends mail on a YP client to `root@podunk-u` if the name of the YP domain is `podunk-u`.

Automatic Forwarding

When an alias (or address) is resolved to the name of a user on the local host, `sendmail` checks for a `.forward` file, owned by the intended recipient, in that user's home directory, and with universal read access. This file can contain one or more addresses or aliases as described above, each of which is sent a copy of the user's mail.

Care must be taken to avoid creating addressing loops in the `.forward` file. When forwarding mail between machines, be sure that the destination machine does not return the mail to the sender through the operation of any YP aliases. Otherwise, copies of the message may "bounce." Usually, the solution is to change the YP alias to direct mail to the proper destination.

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

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

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

FILES

```
/etc/passwd
/etc/aliases
~/.forward
```

SEE ALSO

`newaliases`(1M), `sendmail`(1M), `vacation`(1), `dbm`(3X).

`uucp`(1C), in the *User's Reference Manual.*

NOTES

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

NAME

`syslog.conf` – configuration file for syslogd system log daemon

SYNOPSIS

`/etc/syslog.conf`

DESCRIPTION

The file `/etc/syslog.conf` contains information used by the system log daemon, `syslogd`(1M), to forward a system message to appropriate log files and/or users. `syslog` preprocesses this file through `m4`(1) to obtain the correct information for certain log files.

A configuration entry is composed of two TAB-separated fields:

"selector action"

The *selector* field contains a semicolon-separated list of priority specifications of the form:

facility.level [; *facility.level*]

where *facility* is a system facility, or comma-separated list of facilities, and *level* is an indication of the severity of the condition being logged. Recognized values for *facility* include:

`user`	Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.
`kern`	Messages generated by the kernel.
`mail`	The mail system.
`daemon`	System daemons, such as `ftpd`(1M), `routed`(1M), etc.
`auth`	The authorization system: `login`(1), `su`(1M), `getty`(1M), etc.
`lpr`	The line printer spooling system: `lpr`(1), `lpc`(1M), `lpd`(1M), etc.
`news`	Reserved for the USENET network news system.
`uucp`	Reserved for the UUCP system; it does not currently use the `syslog` mechanism.
`cron`	The `cron` /at facility; `crontab`(1), `at`(1), `cron`(1M), etc.
`local0-7`	Reserved for local use.
`mark`	For timestamp messages produced internally by `syslogd`.
*	An asterisk indicates all facilities except for the `mark` facility.

Recognized values for *level* are (in descending order of severity):

`emerg`	For panic conditions that would normally be broadcast to all users.
`alert`	For conditions that should be corrected immediately, such as a corrupted system database.
`crit`	For warnings about critical conditions, such as hard device errors.
`err`	For other errors.

`warning` For warning messages.

`notice` For conditions that are not error conditions, but may require special handling.

`info` Informational messages.

`debug` For messages that are normally used only when debugging a program.

`none` Do not send messages from the indicated *facility* to the selected file. For example, a *selector* of

```
*.debug;mail.none
```

will send all messages *except* mail messages to the selected file.

The *action* field indicates where to forward the message. Values for this field can have one of four forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the *selector* are to be written to the specified file. The file will be opened in append mode.
- The name of a remote host, prefixed with an `@`, as with: `@`*server*, which indicates that messages specified by the *selector* are to be forwarded to the `syslogd` on the named host.
- A comma-separated list of usernames, which indicates that messages specified by the *selector* are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the *selector* are to be written to all logged-in users.

Blank lines are ignored. Lines for which the first nonwhite character is a '`#`' are treated as comments.

EXAMPLE

With the following configuration file:

```
*.notice;mail.info      /var/log/notice
*.crit                  /var/log/critical
kern,mark.debug         /dev/console
kern.err                @server
*.emerg                 *
*.alert                 root,operator
*.alert;auth.warning    /var/log/auth
```

`syslogd` will log all mail system messages except `debug` messages and all `notice` (or higher) messages into a file named `/var/log/notice`. It logs all critical messages into `/var/log/critical`, and all kernel messages and 20-minute marks onto the system console.

Kernel messages of `err` (error) severity or higher are forwarded to the machine named *server*. Emergency messages are forwarded to all users. The users root and operator are informed of any `alert` messages. All messages from the authorization system of `warning` level or higher are logged in the file `/var/log/auth`.

FILES

```
/etc/syslog.conf
/var/log/notice
/var/log/critical
/var/log/auth
```

SEE ALSO

logger(1), lpr(1), syslogd(1M), syslog(3).

at(1), crontab(1), login(1), lp(1), su(1M) in the *User's Reference Manual.*

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

m4(1) in the *Programmer's Reference Manual.*

SPECIAL FILES (7)

NAME

eqnchar – special character definitions for eqn

SYNOPSIS

eqn /usr/ucblib/pub/eqnchar [*filename*] | troff [*options*]

neqn /usr/ucblib/pub/eqnchar [*filename*] | nroff [*options*]

DESCRIPTION

The eqnchar command contains troff(1) and nroff(1) character definitions for constructing characters that are not available on the Graphic Systems typesetter. These definitions are primarily intended for use with eqn(1) and neqn. It contains definitions for the following characters:

ciplus	⊕	| |	‖	*square*	□
citimes	⊗	*langle*	〈	*circle*	○
wig	~	*rangle*	〉	*blot*	□
-wig	≃	*hbar*	ℏ	*bullet*	•
>wig	≳	*ppd*	⊥	*prop*	∝
<wig	≲	<->	↔	*empty*	∅
=wig	≅	<=>	⇔	*member*	∈
star	∗	|<	≮	*nomem*	∉
bigstar	✳	|>	≯	*cup*	∪
=dot	≐	*ang*	∠	*cap*	∩
orsign	∨	*rang*	∟	*incl*	⊆
andsign	∧	*3dot*	∴	*subset*	⊂
=del	≙	*thf*	∴	*supset*	⊃
oppA	∀	*quarter*	¼	*!subset*	⊄
oppE	∃	*3quarter*	¾	*!supset*	⊅
angstrom	Å	*degree*	°		

FILES

/usr/ucblib/pub/eqnchar

SEE ALSO

eqn(1), nroff(1), troff(1).

NAME

man – macros to format Reference Manual pages

SYNOPSIS

`nroff -man` *filename...*

`troff -man` *filename...*

DESCRIPTION

These macros are used to lay out the reference pages in this manual. Note: if *filename* contains format input for a preprocessor, the commands shown above must be piped through the appropriate preprocessor. This is handled automatically by man(1). See the "Conventions" section.

Any text argument *t* may be zero to six words. Quotes may be used to include SPACE characters in a word. If *text* is empty, the special treatment is applied to the next input line with text to be printed. In this way `.I` may be used to italicize a whole line, or `.SB` may be used to make small bold letters.

A prevailing indent distance is remembered between successive indented paragraphs, and is reset to default value upon reaching a non-indented paragraph. Default units for indents *i* are ens.

Type font and size are reset to default values before each paragraph, and after processing font and size setting macros.

These strings are predefined by -man:

`\*R` '®', '(Reg)' in `nroff`.
`\*S` Change to default type size.

Requests

* n.t.l. = next text line; p.i. = prevailing indent

Request	*Cause Break*	*If no Argument*	*Explanation*
`.B` *t*	no	*t*=n.t.l.*	Text is in bold font.
`.BI` *t*	no	*t*=n.t.l.	Join words, alternating bold and italic.
`.BR` *t*	no	*t*=n.t.l.	Join words, alternating bold and roman.
`.DT`	no	.5i 1i...	Restore default tabs.
`.HP` *i*	yes	*i*=p.i.*	Begin paragraph with hanging indent. Set prevailing indent to *i*.
`.I` *t*	no	*t*=n.t.l.	Text is italic.
`.IB` *t*	no	*t*=n.t.l.	Join words, alternating italic and bold.
`.IP` *x i*	yes	*x*=""	Same as `.TP` with tag *x*.
`.IR` *t*	no	*t*=n.t.l.	Join words, alternating italic and roman.
`.IX` *t*	no	-	Index macro, for Sun internal use.
`.LP`	yes	-	Begin left-aligned paragraph. Set prevailing indent to .5i.
`.PD` *d*	no	*d*=.4v	Set vertical distance between paragraphs.
`.PP`	yes	-	Same as `.LP`.

.RE	yes	-	End of relative indent. Restores prevailing indent.
.RB *t*	no	*t*=n.t.l.	Join words, alternating roman and bold.
.RI *t*	no	*t*=n.t.l.	Join words, alternating roman and italic.
.RS *i*	yes	*i*=p.i.	Start relative indent, increase indent by *i*. Sets prevailing indent to .5i for nested indents.
.SB *t*	no	-	Reduce size of text by 1 point, make text bold.
.SH *t*	yes	-	Section Heading.
.SM *t*	no	*t*=n.t.l.	Reduce size of text by 1 point.
.SS *t*	yes	*t*=n.t.l.	Section Subheading.
.TH *n s d f m*	yes	-	Begin reference page *n*, of of section *s*; *d* is the date of the most recent change. If present, *f* is the left page footer; *m* is the main page (center) header. Sets prevailing indent and tabs to .5i.
.TP *i*	yes	*i*=p.i.	Begin indented paragraph, with the tag given on the next text line. Set prevailing indent to *i*.
.TX *t p*	no	-	Resolve the title abbreviation *t*; join to punctuation mark (or text) *p*.

Conventions

When formatting a manual page, man examines the first line to determine whether it requires special processing. For example a first line consisting of:

```
'\" t
```

indicates that the manual page must be run through the tbl(1) preprocessor.

A typical manual page for a command or function is laid out as follows:

.TH *title* [1-8]

The name of the command or function, which serves as the title of the manual page. This is followed by the number of the section in which it appears.

.SH NAME

The name, or list of names, by which the command is called, followed by a dash and then a one-line summary of the action performed. All in roman font, this section contains no troff(1) commands or escapes, and no macro requests. It is used to generate the whatis(1) database.

.SH SYNOPSIS

Commands:

The syntax of the command and its arguments, as typed on the command line. When in boldface, a word must be typed exactly as printed. When in italics, a word can be replaced with an argument that you supply. References to bold or italicized items are not capitalized in other sections, even when they begin a sentence.

Syntactic symbols appear in roman face:

[ ] An argument, when surrounded by brackets is optional.

| Arguments separated by a vertical bar are exclusive. You can supply only one item from such a list.

. . . Arguments followed by an elipsis can be repeated. When an elipsis follows a bracketed set, the expression within the brackets can be repeated.

Functions:

If required, the data declaration, or `#include` directive, is shown first, followed by the function declaration. Otherwise, the function declaration is shown.

`.SH DESCRIPTION`

A narrative overview of the command or function's external behavior. This includes how it interacts with files or data, and how it handles the standard input, standard output and standard error. Internals and implementation details are normally omitted. This section attempts to provide a succinct overview in answer to the question, "what does it do?"

Literal text from the synopsis appears in constant width, as do literal filenames and references to items that appear elsewhere in the reference manuals. Arguments are italicized.

If a command interprets either subcommands or an input grammar, its command interface or input grammar is normally described in a USAGE section, which follows the OPTIONS section. The DESCRIPTION section only describes the behavior of the command itself, not that of subcommands.

`.SH OPTIONS`

The list of options along with a description of how each affects the command's operation.

`.SH FILES`

A list of files associated with the command or function.

`.SH SEE ALSO`

A comma-separated list of related manual pages, followed by references to other published materials.

`.SH DIAGNOSTICS`

A list of diagnostic messages and an explanation of each.

`.SH BUGS`

A description of limitations, known defects, and possible problems associated with the command or function.

FILES

`/usr/ucblib/doctools/man`

SEE ALSO

man(1), nroff(1), troff(1), whatis(1)

NAME

me – macros for formatting papers

SYNOPSIS

`nroff -me` [*options*] *filename* ...
`troff -me` [*options*] *filename* ...

DESCRIPTION

This package of `nroff` and `troff` macro definitions provides a canned formatting facility for technical papers in various formats. When producing 2-column output on a terminal, filter the output through *col*(1).

The macro requests are defined below. Many `nroff` and `troff` requests are unsafe in conjunction with this package, however, these requests may be used with impunity after the first .pp:

`.bp`	begin new page
`.br`	break output line here
`.sp n`	insert n spacing lines
`.ls n`	(line spacing) n=1 single, n=2 double space
`.na`	no alignment of right margin
`.ce n`	center next n lines
`.ul n`	underline next n lines
`.sz +n`	add n to point size

Output of the `eqn`, `meqn`, `mefer`, and `tbl`(1) preprocessors for equations and tables is acceptable as input.

REQUESTS

In the following list, initialization refers to the first `.pp`, `.lp`, `.ip`, `.np`, `.sh`, or `.uh` macro. This list is incomplete.

Request	Initial Value	Cause Break	Explanation
`.(c`	-	yes	Begin centered block
`.(d`	-	no	Begin delayed text
`.(f`	-	no	Begin footnote
`.(l`	-	yes	Begin list
`.(q`	-	yes	Begin major quote
`.(x`*x*	-	no	Begin indexed item in index *x*
`.(z`	-	no	Begin floating keep
`.)c`	-	yes	End centered block
`.)d`	-	yes	End delayed text
`.)f`	-	yes	End footnote
`.)l`	-	yes	End list
`.)q`	-	yes	End major quote
`.)x`	-	yes	End index item
`.)z`	-	yes	End floating keep
`.++` *m H*	-	no	Define paper section. *m* defines the part of the paper, and can be `C` (chapter), `A` (appendix), `P` (preliminary, for instance, abstract, table of contents, etc.), `B` (bibliography), `RC` (chapters renumbered from page one each chapter), or `RA` (appendix renumbered from page one).

.+c *T*	-	yes	Begin chapter (or appendix, etc., as set by .++). *T* is the chapter title.
.1c	1	yes	One column format on a new page.
.2c	1	yes	Two column format.
.EN	-	yes	Space after equation produced by eqn or meqn.
.EQ *x y*	-	yes	Precede equation; break out and add space. Equation number is *y*. The optional argument *x* may be *I* to indent equation (default), *L* to left-adjust the equation, or *C* to center the equation.
.GE	-	yes	End *gremlin* picture.
.GS	-	yes	Begin *gremlin* picture.
.PE	-	yes	End pic picture.
.PS	-	yes	Begin pic picture.
.TE	-	yes	End table.
.TH	-	yes	End heading section of table.
.TS *x*	-	yes	Begin table; if *x* is *H* table has repeated heading.
.ac *A N*	-	no	Set up for ACM style output. *A* is the Author's name(s), *N* is the total number of pages. Must be given before the first initialization.
.b *x*	no	no	Print *x* in boldface; if no argument switch to boldface.
.ba +*n*	0	yes	Augments the base indent by *n*. This indent is used to set the indent on regular text (like paragraphs).
.bc	no	yes	Begin new column
.bi *x*	no	no	Print *x* in bold italics (nofill only)
.bu	-	yes	Begin bulleted paragraph
.bx *x*	no	no	Print *x* in a box (nofill only).
.ef *'x'y'z*	''''	no	Set even footer to *x y z*
.eh *'x'y'z*	''''	no	Set even header to *x y z*
.fo *'x'y'z*	''''	no	Set footer to *x y z*
.hx	-	no	Suppress headers and footers on next page.
.he *'x'y'z*	''''	no	Set header to x y z
.hl	-	yes	Draw a horizontal line
.i *x*	no	no	Italicize *x*; if *x* missing, italic text follows.
.ip *x y*	no	yes	Start indented paragraph, with hanging tag *x*. Indentation is *y* ens (default 5).
.lp	yes	yes	Start left-blocked paragraph.
.lo	-	no	Read in a file of local macros of the form .*x. Must be given before initialization.
.np	1	yes	Start numbered paragraph.
.of *'x'y'z*	''''	no	Set odd footer to x y z
.oh *'x'y'z*	''''	no	Set odd header to x y z
.pd	-	yes	Print delayed text.
.pp	no	yes	Begin paragraph. First line indented.
.r	yes	no	Roman text follows.
.re	-	no	Reset tabs to default values.
.sc	no	no	Read in a file of special characters and diacritical marks. Must be given before initialization.
.sh *n x*	-	yes	Section head follows, font automatically bold. *n* is level of section, *x* is title of section.

.sk	no	no	Leave the next page blank. Only one page is remembered ahead.
.sm *x* -	*no*	*Set x in a smaller pointsize.*	
.sz +*n*	10p	no	Augment the point size by *n* points.
.th	no	no	Produce the paper in thesis format. Must be given before initialization.
.tp	no	yes	Begin title page.
.u *x*	-	no	Underline argument (even in troff). (Nofill only).
.uh	-	yes	Like .sh but unnumbered.
.xp *x*	-	no	Print index *x*.

FILES

/usr/ucblib/doctools/tmac/e
/usr/ucblib/doctools/tmac/*.me

SEE ALSO

eqn(1), nroff(1), troff(1), refer(1), tbl(1)

NAME

ms – text formatting macros

SYNOPSIS

`nroff -ms` [*options*] *filename* ...

`troff -ms` [*options*] *filename* ...

DESCRIPTION

This package of `nroff`(1) and `troff`(1) macro definitions provides a formatting facility for various styles of articles, theses, and books. When producing 2-column output on a terminal or lineprinter, or when reverse line motions are needed, filter the output through `col`(1V). All external `-ms` macros are defined below.

Note: this `-ms` macro package is an extended version written at Berkeley and is a superset of the standard `-ms` macro packages as supplied by Bell Labs. Some of the Bell Labs macros have been removed; for instance, it is assumed that the user has little interest in producing headers stating that the memo was generated at Whippany Labs.

Many `nroff` and `troff` requests are unsafe in conjunction with this package. However, the first four requests below may be used with impunity after initialization, and the last two may be used even before initialization:

`.bp`	begin new page
`.br`	break output line
`.sp` *n*	insert n spacing lines
`.ce` *n*	center next n lines
`.ls` *n*	line spacing: *n*=1 single, *n*=2 double space
`.na`	no alignment of right margin

Font and point size changes with `\f` and `\s` are also allowed; for example, `\fIword\fR` will italicize *word*. Output of the `tbl`(1), `eqn`(1) and `refer`(1) preprocessors for equations, tables, and references is acceptable as input.

REQUESTS

Macro Name	Initial Value	Break? Reset?	Explanation
`.AB` *x*	–	y	begin abstract; if *x*=no do not label abstract
`.AE`	–	y	end abstract
`.AI`	–	y	author's institution
`.AM`	–	n	better accent mark definitions
`.AU`	–	y	author's name
`.B` *x*	–	n	embolden *x*; if no *x*, switch to boldface
`.B1`	–	y	begin text to be enclosed in a box
`.B2`	–	y	end boxed text and print it
`.BT`	date	n	bottom title, printed at foot of page
`.BX` *x*	–	n	print word *x* in a box
`.CM`	if t	n	cut mark between pages
`.CT`	–	y,y	chapter title: page number moved to CF (TM only)
`.DA` *x*	if n	n	force date *x* at bottom of page; today if no *x*
`.DE`	–	y	end display (unfilled text) of any kind
`.DS` *x y*	I	y	begin display with keep; *x*=I, L, C, B; *y*=indent

.ID *y*	8n,.5i	y	indented display with no keep; *y*=indent
.LD	–	y	left display with no keep
.CD	–	y	centered display with no keep
.BD	–	y	block display; center entire block
.EF *x*	–	n	even page footer *x* (3 part as for .tl)
.EH *x*	–	n	even page header *x* (3 part as for .tl)
.EN	–	y	end displayed equation produced by eqn
.EQ *x y*	–	y	break out equation; *x*=L,I,C; *y*=equation number
.FE	–	n	end footnote to be placed at bottom of page
.FP	–	n	numbered footnote paragraph; may be redefined
.FS *x*	–	n	start footnote; *x* is optional footnote label
.HD	undef	n	optional page header below header margin
.I *x*	–	n	italicize *x*; if no *x*, switch to italics
.IP *x y*	–	y,y	indented paragraph, with hanging tag *x*; *y*=indent
.IX *x y*	–	y	index words *x y* and so on (up to 5 levels)
.KE	–	n	end keep of any kind
.KF	–	n	begin floating keep; text fills remainder of page
.KS	–	y	begin keep; unit kept together on a single page
.LG	–	n	larger; increase point size by 2
.LP	–	y,y	left (block) paragraph.
.MC *x*	–	y,y	multiple columns; *x*=column width
.ND *x*	if t	n	no date in page footer; *x* is date on cover
.NH *x y*	–	y,y	numbered header; *x*=level, *x*=0 resets, *x*=S sets to *y*
.NL	10p	n	set point size back to normal
.OF *x*	–	n	odd page footer *x* (3 part as for .tl)
.OH *x*	–	n	odd page header *x* (3 part as for .tl)
.P1	if TM	n	print header on first page
.PP	–	y,y	paragraph with first line indented
.PT	- % -	n	page title, printed at head of page
.PX *x*	–	y	print index (table of contents); *x*=no suppresses title
.QP	–	y,y	quote paragraph (indented and shorter)
.R	on	n	return to Roman font
.RE	5n	y,y	retreat: end level of relative indentation
.RP *x*	–	n	released paper format; *x*=no stops title on first page
.RS	5n	y,y	right shift: start level of relative indentation
.SH	–	y,y	section header, in boldface
.SM	–	n	smaller; decrease point size by 2
.TA	8n,5n	n	set TAB characters to 8n 16n ... (nroff) 5n 10n ... (troff)
.TC *x*	–	y	print table of contents at end; *x*=no suppresses title
.TE	–	y	end of table processed by tbl
.TH	–	y	end multi-page header of table
.TL	–	y	title in boldface and two points larger
.TM	off	n	UC Berkeley thesis mode
.TS *x*	–	y,y	begin table; if *x*=H table has multi-page header
.UL *x*	–	n	underline *x*, even in troff
.UX *x*	–	n	UNIX; trademark message first time; *x* appended
.XA *x y*	–	y	another index entry; *x*=page or no for none; *y*=indent
.XE	–	y	end index entry (or series of .IX entries)
.XP	–	y,y	paragraph with first line exdented, others indented

.XS *x y*	–	y	begin index entry; *x*=page or no for none; *y*=indent
.1C	on	y,y	one column format, on a new page
.2C	–	y,y	begin two column format
.]–	–	n	beginning of `refer` reference
.[0	–	n	end of unclassifiable type of reference
.[N	–	n	N= 1:journal-article, 2:book, 3:book-article, 4:report

REGISTERS

Formatting distances can be controlled in `-ms` by means of built-in number registers. For example, this sets the line length to 6.5 inches:

```
.nr LL 6.5i
```

Here is a table of number registers and their default values:

Name	Register Controls	Takes Effect	Default
PS	point size	paragraph	10
VS	vertical spacing	paragraph	12
LL	line length	paragraph	6i
LT	title length	next page	same as LL
FL	footnote length	next .FS	5.5i
PD	paragraph distance	paragraph	1v (if n), .3v (if t)
DD	display distance	displays	1v (if n), .5v (if t)
PI	paragraph indent	paragraph	5n
QI	quote indent	next .QP	5n
FI	footnote indent	next .FS	2n
PO	page offset	next page	0 (if n), ~1i (if t)
HM	header margin	next page	1i
FM	footer margin	next page	1i
FF	footnote format	next .FS	0 (1, 2, 3 available)

When resetting these values, make sure to specify the appropriate units. Setting the line length to 7, for example, will result in output with one character per line. Setting FF to 1 suppresses footnote superscripting; setting it to 2 also suppresses indentation of the first line; and setting it to 3 produces an .IP-like footnote paragraph.

Here is a list of string registers available in `-ms`; they may be used anywhere in the text:

Name	String's Function
*Q	quote (" in nroff, `` in troff)
*U	unquote (" in nroff, '' in troff)
*–	dash (-- in nroff, — in troff)
*(MO	month (month of the year)
*(DY	day (current date)
**	automatically numbered footnote
*´	acute accent (before letter)
*`	grave accent (before letter)
*^	circumflex (before letter)
*,	cedilla (before letter)
*:	umlaut (before letter)
*~	tilde (before letter)

When using the extended accent mark definitions available with .AM, these strings should come after, rather than before, the letter to be accented.

FILES

/usr/ucb/lib/doctools/tmac/s
/usr/ucblib/doctools/tmac/ms.???

SEE ALSO

col(1V), eqn(1), nroff(1), refer(1), tbl(1), troff(1)

BUGS

Floating keeps and regular keeps are diverted to the same space, so they cannot be mixed together with predictable results.

BSD Subject Index

F

G

H

I

K

L

M

N

P

R

S

T

U

V

W

XENIX Compatibility Guide

Overview

This guide describes the new commands and system calls in the XENIX Compatibility Package that allow System V Release 4.0 users to run programs developed in the XENIX environment. This guide is organized into the following sections:

1. A brief description of commands by function.
2. A brief description of system calls by function.
3. A section containing the manual pages of the listed features.

Audience

This compatibility guide is useful for programmers who need specific information about a feature's functionality and system administrators who are involved with file management.

Running XENIX Programs

Various modifications have been made to the UNIX kernel to make System V Release 4.0 compatible with XENIX. Before running any shell scripts or programs, please note the following:

- Read the "Tunable Parameters" section to learn which XENIX parameters are defined in the `/etc/master.d/kernel` file.
- In the *Source Code Provision Build Instructions,* read the section on installing the XENIX Compatibility Package and on rebuilding the kernel to give System V the ability to allocate IPC semaphores and shared data.
- Once the compatibility package is installed, you can make use of it by setting your `PATH` variable so that `/usr/ucb` comes before the default System V path directories such as `/usr/bin` and `/usr/bin`. `/usr/ucb` contains the compatibility package commands.

Tunable Parameters

Tunable system parameters are used to set various table sizes and system thresholds to handle the expected system load. Caution should be used when changing these variables since such changes can directly affect system performance.

The following table gives the tunable parameters for XENIX compatibility.

Parameter	default	min	max	size
DSTFLAG	0	0	1	-
TIMEZONE	0	0	1440	-
XSEMMAX	60	20	60	12
XSDSEGS	25	1	25	12
XSDSLOTS	3	1	3	20 x XSDSEGS

Timezone Parameters

The following tunable parameters are defined in `/etc/master.d/kernel`. They are defined for the `ftime`(2) system call. Use of the `TZ` environment variable is encouraged instead.

`Dstflag` Specifies whether daylight savings time applies locally during the appropriate part of the year. The default is 0.

`Timezone` Specifies the local time zone measured in minutes of time westward from Greenwich The default is 0.

XENIX Shared Data Parameters

The following tunable parameters are defined in `/etc/master.d/kernel`.

`XSDSEGS` Specifies the number of shared data segments in the system. The minumum value is 1, and the default is 25.

`XSDSLOTS` (XSDSEGS x XSDSLOTS) specifies the maximum number of shared data segment attachments allowed in the system. The minumum value of XSDSLOTS is 1, and the default is 3.

XENIX Semaphore Tunable Parameters

The following parameters are defined in `/etc/master.d/xnamfs`.

`XSEMMAX` Specifies the number of XENIX semaphores in the system. The default value is 60, and the mininum is 1.

Refer to the "Tunable Parameters" section in the "Performance Management" chapter of the *System Administrator's Guide* for more information.

Commands by Function

UNIX/XENIX Administration

`backup`	This is a shell script that invokes `find` and `cpio` to do backups. The backup consists of a header record, some bit mask records, a group of records describing file system directories, a group of records describing file system files, and some records describing a second bit mask.
`custom`	Installs selected parts of the XENIX system.
`restore`	Restore file to original directory.
`xinstall`	This command installs the XENIX system.

File Management

`copy`	This command copies contents of directories to another directory. Since there may be more than one source directory, it effectively allows you to copy whole file systems.
`egrep`	Added the `-y` option to ignore upper/lower case distinctions.
`fgrep`	Added the `-y` option to ignore upper/lower case distinctions.
`fixperm`	Corrects or initializes the permissions and ownership on files.
`grep`	Added the `-y` option to ignore upper/lower case distinctions.
`hd`	Displays the contents of files in hexadecimal, octal, decimal, and character formats.
`ls`	Lists the contents of files and directories in columns.
`settime`	Sets the access and modification dates for one or more files.

Terminal Management

`tset`	Allows the user to set a terminal's `ERASE` and `kill` characters, and define the terminal's type and capabilities by creating values for the `TERM` and `TERMCAP` environment variables.

Shell Program Utilities

`random`	Generates a random number, which, by default, is 0 or 1.
`yes`	Repeatedly outputs the 'y' character.

System Calls

File Management

`chsize`	Changes the size of the file to a specified length by either truncating the file or padding it with an appropriate number of bytes.
`locking`	Allows a specified number of bytes in a file to be controlled by the locking process, to lock or unlock a file region for reading or writing.
`mknod`	Creates a new file name by the pathname pointed to by *path*.
`rdchk`	Checks to see if a process will block if it attempts to read the data in a file.
`stat`	Obtains information about the named file pointed to by *path*.

Process Management

`lock`	Locks a process in primary memory.
`nap`	Suspends the execution of a current process for a short interval.
`ftime`	Gets the time and date and returns the time in a structure pointed to by its argument.

Inter-Process Communications

`creatsem`	Defines a binary semaphore and returns a unique semaphore number used by the `waitsem` and `sigsem` system calls to set parameters. `creatsem` also manages mutually exclusive access to a resource, shared variable or critical section of a program.
`nbwaitsem`	Will fail if a semaphore is busy instead of waiting [see `waitsem`(2)].

`opensem`	Opens a named semaphore and returns the unique semaphore semaphore identification number used by `waitsem` and `sigsem`.
`stat`	Puts data returned by `stat` system call in a special format.
`sdenter`	Synchronizes access to a shared data segment by indicating that the current process is about to access the contents of the same.
`sdfree`	Detaches the current process from the shared data segment that is attached at the specified address [see `sdget`(2)].
`sdget`	Attaches a shared data segment to the data space of the current process.
`sdgetv`	Synchronizes cooperating processes that are using shared data segments.
`sdleave`	Used in conjunction with `sdenter` to synchronize processes using shared data segments [see `sdenter`(2)].
`sdwaitv`	Used in conjunction with `sdgetv` to synchronize processes using shared data segments [see `sdgetv`(2)].
`sigsem`	Signals a process that is waiting on the semaphore that it may proceed and use the resource governed by the semaphore.
`waitsem`	Awaits and checks access to a resource governed by a semaphore. It is used in conjunction with `sigsem`.

Table of Contents

1. Commands

2. System Calls

4. File Formats

Permuted Index

NAME

backup – performs backup functions

SYNOPSIS

backup [-t] [-p | -c | -f <files> | -u "<user1> [user2]"] -d <device>
backup -h

DESCRIPTION

-h produces a history of backups. Tells the user when the last complete and incremental/partial backups were done.

-c complete backup. All files changed since the system was installed are backed up. If an incremental/partial backup was done, all files modified since that time are backed up, otherwise all files modified since the last complete backup are backed up. A complete backup must be done before a partial backup.

-f backup files specified by the *<files>* argument. file names may contain characters to be expanded (i.e., *, .) by the shell. The argument must be in quotes.

-u backup a user's home directory. All files in the user's home directory will be backed up. At least one user must be specified but it can be more. The argument must be in quotes if more than one user is specified. If the user name is "all", then all the user's home directories will be backed up.

-d used to specify the device to be used. It defaults to /dev/SA/diskette.

-t used when the device is a tape. This option must be used with the -d option when the tape device is specified.

A complete backup must be done before a partial backup can be done. Raw devices rather than block devices should always be used. The program can handle multi-volume backups. The program will prompt the user when it is ready for the next medium. The program will give you an estimated number of floppies/tapes that will be needed to do the backup. Floppies must be formatted before the backup is done. Tapes do not need to be formatted. If backup is done to tape, the tape must be rewound.

NAME

copy - copies groups of files.

SYNOPSIS

copy [*option*]...*source...dest*

DESCRIPTION

The copy command copies the contents of directories to another directory. It is possible to copy whole file systems since directories are made when needed.

If files, directories, or special files do not exist at the destination, then they are created with the same modes and flags as the source. In addition, the super-user may set the user and group ID. The owner and mode are not changed if the destination file exists. Note that there may be more than one source directory. If so, the effect is the same as if the copy command had been issued for each source directory with the same destination directory for each copy.

All of the options must be given as separate arguments, and they may appear in any order even after the other arguments. The arguments are:

`-a` Asks the user before attempting a copy. If the response does not begin with a "y", then a copy is not done. This option also sets the ad option.

`-l` Uses links instead whenever they can be used. Otherwise a copy is done. Note that links are never done for special files or directories.

`-n` Requires the destination file to be new. If not, then the copy command does not change the destination file. The -n flag is meaningless for directories. For special files an -n flag is assumed (i.e., the destination of a special file must not exist).

`-o` If set then every file copied has its owner and group set to those of source. If not set, then the file's owner is the user who invoked the program.

`-m` If set, then every file copied has its modification time and access time set to that of the source. If not set, then the modification time is set to the time of the copy.

`-r` If set, then every directory is recursively examined as it is encountered. If not set, then any directories that are found are ignored.

`-ad` Asks the user whether an -r flag applies when a directory is discovered. If the answer does not begin with a "y", then the directory is ignored.

`-v` If the verbose option is set, messages are printed that reveal what the program is doing.

source This may be a file, directory or special file. It must exist. If it is not a directory, then the results of the command are the same as for the cp command.

dest The destination must be either a file or directory that is different from the source. If *source* and *destination* are anything but directories, then copy acts just like a cp command. If both are directories, then copy copies each file into the destination directory according to the flags that have been set.

NOTES

Special device files can be copied. When they are copied, any data associated with the specified device is not copied.

NAME

custom - install specific portions of a UNIX package

SYNOPSIS

custom [-ir] [package]] [-m device] [-f [file]]

DESCRIPTION

custom allows the super-user to create a custom installation by selectively installing or deleting portions of the UNIX packages to or from the 386 operating system. It can be used interactively or it can be invoked from the command line with applicable command options.

Files are extracted or deleted in *packages*. A *package* is a collection of individual files that are grouped together in *sets*.

When in interactive mode, custom prompts you for volume 1 of the new product distribution and extracts the product information necessary to support it. The following menu provides support for adding or removing a package:

1. Install one or more packages
2. Remove one or more packages
3. List the files in a package
4. Install a single file
5. Select a new set to customize
6. Display current disk usage
7. Help

When you enter a menu option, you are prompted for further information. The following describes what actions are necessary for each menu option:

1. Install

   Prompts for one or more package names

   Calculates which installation volumes (distribution media) are needed and then prompts the user for the correct volume numbers. If multiple packages are specified, the names should be separated by spaces on the command line.

This option, as well as "2" and "3," displays a list of available packages in the selected set. Each line describes the package name, whether the package is fully installed, not installed or partially installed, the size of the package (in 512 byte blocks), and a one line description of the package contents.

2. Remove

   Prompts for one or more package names.

   Deletes the correct files in the specified package. If multiple packages are specified, the names should be separated by spaces on the command line.

   Displays available packages (see option "1").

3. List files in a package

   Lists all files in the specified package

   Prompts for one or more package names. Enter the name of the desired package(s).

   Displays available packages (see option "1").

4. Install a single file
   Retrieves the specified file from the distribution set

   Filename should be a full pathname relative to the root directory "/".
5. Select a new set
   Allows the user to work from a different set
6. Display current disk usage
   Tells current disk usage.
7. Help
   Prints a page of instructions to help you use `custom`.

`-s` A set identifier

`-i` Install the specified package(s)

`-r` Remove the specified package(s)

`-l` List the files in the specified package(s)

`-f` Install the specified file

FILES

`/etc/perms/*`

SEE ALSO

`fixperm`(1M), `df`(1M), `du`(1M), `install`(1M)

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.

-h Suppresses printing of filenames when searching multiple files.

-i, -y
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).

NOTES

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

NAME

fgrep – search a file for a character string

SYNOPSIS

fgrep [options] string [file ...]

DESCRIPTION

fgrep (fast grep) seaches 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.

-h Suppresses printing of filenames when searching multiple files.

-i, -y
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

fixperm - corrects or initializes file permissions and ownership.

SYNOPSIS

fixperm [-cDjilnSsvw[-d *package*] [-u *package*]] *specfile*

DESCRIPTION

For each line in the specification file *specfile*, fixperm makes the listed pathname conform to a specification. fixperm is typically used to configure a XENIX system upon installation. Nonsuper-users can only use fixperm with the -D, -f, -l, or -n options. Only super-users can use the -c, -d, -i, -n, -S, -u, -v, and -w options.

The following options are available:

Option	Description
-c	Creates empty files and missing directories.
-D	Lists directories only on standard output. Does not modify target files.
-d *package*	Processes input lines beginning with given package specifier string (see above). For instance, -dBASE processes only items specified as belonging to the Basic utilities set. The default action is to process all lines.
-f	Lists files only on standard output. Does not modify target files.
-i	Checks only if the selected packages are installed. Return values are: 0: package completely installed 4: package not installed 5: package partially installed
-l	Lists files and directories on standard output. Does not modify target files.
-n	Reports errors only. Does not modify target files.
-S	Issues a complaint if files are not in x.out format.
-s	Modifies special device files in addition to the rest of the permlist.
-u *package*	Causes similar action to -d option, but processes items that are not part of the given package.
-v	Issues a complaint if executable files are: 1) word-swapped 2) not fixed-stack 3) not separate I and D 4) not stripped
-w	Lists location (volume number) of the specified files or directories.

Specification File Format

Each nonblank line in the specification file consists of either a comment or an item specification. A comment is any text from a pound sign "#" up to the end of the line. There is one item specification per line. User and group id numbers must be specified at the top of the specification file for each user and group mentioned in the file.

An item specification consists of a package specifier, a permission specification, owner and group specifications, the number of links on the file, the filename, and an optional volume number.

The package specifier is an arbitrary string that is the name of a package within a distribution set. A package is a set of files.

A permission specification follows the package specifier. The permission specification consists of a file type, followed by a numeric permission specification. The item specification is one of the following characters:

Character	Description
x	executable
a	archive
e	empty file (create if -c option given)
b	block device
c	character device
d	directory
f	text file
p	named pipe

If the item specification is given as an uppercase letter, the file associated with it is optional, and `fixperm` will not return an error message if it does not exist.

The numeric permission conforms to the scheme described in chmod. The owner and group permissions are in the third column separated by slash, such as "bin/bin". The fourth column indicates the number of links. If there are links to the file, the next line contains the linked filename with no other information. The fifth column is a pathname. The pathname must be relative (not preceded by a slash "/"). The sixth column is only used for special files, major and minor device numbers, or volume numbers.

Examples

The following two lines make a distribution and invoke `tar` to archive only the files in `base.perms` on `/dev/sample`:

```
/etc/fixperm -f/etc/base.perms>list
    tar cfF/dev/sample list
```

This command line reports BASE package errors:

```
/etc/fixperm-nd BASE
```

NOTES

`fixperm` is usually only run by a shell script at installation.

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, -y
Ignore upper/lower case distinction during comparisons.

-h Suppresses printing of filenames when searching multiple files.

-l Print the names of files with matching lines once, separated by new-lines. Does not repeat the names of files when the pattern is found more than once.

-n Precede each line by its line number in the file (first line is 1).

-s 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

hd - displays files in hexadecimal format.

SYNOPSIS

hd [*-format*[-s *offset*][-n *count*][*file*]

DESCRIPTION

The hd command displays the contents of files in hexadecimal octal, decimal and character formats. Control over the specification of ranges of characters is also available. The default behavior is with the following flags set: "-abx -A". This says that addresses (file offsets) and bytes are printed in hexadecimal and that characters are also printed. If no *file* argument is given, the standard input is read.

Options include: Specify the beginning offset in the file where printing is to begin. If no 'file' argument is given, or if a seek fails because the input is a pipe, 'offset' bytes are read from the input and discarded. Otherwise, a seek error will terminate processing of the current file.

The *offset* may be given in decimal, hexadecimal (preceded by '0x'), or octal (preceded by a '0'). It is optionally followed by one of the following multipliers: w, l, b, or k; for words (2 bytes), long words (4 bytes), blocks (512 bytes), or K bytes (1024 bytes). Note that this is the one case where "b" does not stand for bytes. Since specifying a hexadecimal offset in blocks would result in an ambiguous trailing 'b', any offset and multiplier may be separated by an asterisk (*). Specify the number of bytes to process. The *count* is in the same format as *offset*, above.

Format Flags

Format flags may specify addresses, characters, bytes, words (2 bytes), or longs (4 bytes) to be printed in hexadecimal, decimal, or octal. Two special formats may also be indicated: test or ASCII. Format and base specifiers amy be freely combined and repeated as desired in order to specify different bases (hexadecimal, decimal or octal) for different output formats (addresses, characters, etc.). All format flags appearing in a single argument are applied as appropriate to all other flags in that argument.

Output format specifiers for address, characters, bytes, words, longs and ASCII, respectively. Only one base specifier will be used for addresses; the address will appear on the first line of output that begins each new offset in the input.

The character format prints printable characters unchanged, special C escapes as defined in the language, and remaining values in the specified base.

The ASCII format prints all printable characters unchanged, and all others as a period (.). This format appears to the right of the first of other specified output formats. A base specifier has no meaning with the SCII format. If no other output format (other than addresses) is given, bx is assumed. If no base specifier is given, all of xdo are used. Output base specifiers for hexadecimal, decimal and octal. If no format specifier is given, all of acbwl are used. Print a test file, each line preceded by the address in the file. Normally, lines should be terminated by a \n character; but long lines will be broken up. Control characters in the range 0x00 to 0x1f are rpinted as '^@' to '^_'. Bytes with the high bit set are preceded by a tilde (~) and printed as if the high bit were not set. The special characters

(^,~,) are preceded by a backslash ( ) to escape their special meaning. As special cases, two values are represented numerically as ^177' and ^377'. This flag will override all output format specifiers except addresses.

NAME

ls, lc – list contents of directory

SYNOPSIS

ls [-RadLCxmlnogrtucpFbqisfl] [*names*]
lc [-1CFLRabcfgilmnopqrstux] [*name...*]

DESCRIPTION

For each directory argument, ls lists the contents of the directory for each file argument. lc functions the same as ls except that the lc default output format is columnar, even if the output is redirected. ls repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The default format for output directed to a terminal is multi-column with entries sorted down the columns. The -1 option allows single column output and -m 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.

-L If an argument is a symbolic link, list the file or directory the link references rather than the link itself.

-C Multi-column output with entries sorted down the columns. This is the default output format.

-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 instead contains the major and minor device numbers rather than a size. If the file is a symbolic link, the filename is printed followed by "->" and the pathname of the referenced file.

-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 the file is a directory.

-F Put a slash (/) after each filename if the file is a directory, an asterisk (*) if the file is an executable, and an ampersand (@) if the file is a symbolic link.

-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.

-1 Print one entry per line of output.

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;
l the entry is a symbolic link;
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.
s the entry is a XENIX semaphore.
m the entry is a XENIX shared data (memory).

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 `l`ong list) prints its output as follows:

```
-rwxrwxrwx  1 smith   dev     10876  May 16 9:42 part2
```

Reading from right to left, you see that the current directory holds one file, named `part2`. Next, the last time that file's contents were modified was 9:42 A.M. on May 16. The file contains 10,876 characters, or bytes. The owner of the file, or the user, belongs to the group `dev` (perhaps indicating "development"), and his or her login name is `smith`. The number, in this case `1`, indicates the number of links to file `part2`; see `cp`(1). Finally, the dash and letters tell you that user, group, and others have permissions to read, write, and execute `part2`.

The execute (`x`) symbol here occupies the third position of the three-character sequence. A `-` in the third position would have indicated a denial of execution permissions.

The permissions are indicated as follows:

- `r` the file is readable
- `w` the file is writable
- `x` the file is executable
- `-` the indicated permission is *not* granted
- `l` mandatory locking occurs 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 you login as.

In the case of the sequence of group permissions, `l` may occupy the third position. `l` refers to mandatory file and record `l`ocking. 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 prints 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 provides information on all files, including those that begin with a dot (`a`), the `i`-number—the memory address of the i-node associated with the file—printed in the left-hand column (`i`); the size (in blocks) of the files, printed in the column to the right of the i-numbers (`s`); finally, the report is displayed in the numeric version of the long list, printing the `UID` (instead of user name) and `GID` (instead of group name) numbers associated with the files.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

FILES

`/etc/passwd`	user IDs for `ls -l` and `ls -o`
`/etc/group`	group IDs for `ls -l` and `ls -g`
`/usr/share/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 *System Administrator's Guide*.

Unprintable characters in file names may confuse the columnar output options.

NAME

random - generates a random number.

SYNOPSIS

random [*-s*][*scale*]

DESCRIPTION

random generates a random number, *scale*, on the standard output, and returns the number as its exit value. By default, this number is either 0 or 1. If *scale* is given a value between 1 and 255, then the range of the random value is from 0 to *scale*. If *scale* is greater than 255, an error message is printed.

When the **-s** (silent) option is given, then the random number is returned as an exit value, but is not printed on the standard output. If an error occurs, random returns an exit value of zero.

SEE ALSO

rand(3C).

NOTES

This command does not perform any floating point computations. random uses the time of day as a seed.

NAME

restore – restore file to original directory

SYNOPSIS

restore [-c] [-i] [-o] [-t] [-d <*device*>] | [pattern [pattern]...]

DESCRIPTION

-c complete restore. All files on the tape are restored.

-i gets the index file off of the medium. This only works when the archive was created using backup. The output is a list of all the files on the medium. No files are actually restored.

-o overwrite existing files. If the file being restored already exists it will not be restored unless this option is specified.

-t indicates that the tape device is to be used. Must be used with the -d option when restoring from tape.

-d <*device*> is the raw device to be used. It defaults to dev/SA/diskette

When doing a restore, one or more patterns can be specified. These patterns are matched against the files on the tape. When a match is found, the file is restored. Since backups are done using full pathnames, the file is restored to its original directory. Metacharacters can be used to match multiple files. The patterns should be in quotes to prevent the characters from being expanded before they are passed to the command. If no patterns are specified, it defaults to restoring all files. If a pattern does not match any file on the tape, a message is printed.

When end of medium is reached, the user is prompted for the next media. The user can exit at this point by typing q. (This may cause files to be corrupted if a file happens to span a medium.) In general, quitting in the middle is not a good idea.

If the file already exists and an attempt is made to restore it without the -o option, the file name will be printed on the screen followed by a question mark. This file will not be retored.

In order for multi-volume restores to work correctly, the raw device must be used.

SEE ALSO

sh(1).

NAME

settime - changes the access and modification dates of files.

SYNOPSIS

settime *mmddhhmm*[*yy*] [-f *fname*] *name...*

DESCRIPTION

Sets the access and modification dates for one or more files. The dates are set to the specified date, or to the access and modification dates of the file specified via -f. Exactly one of these methods must be used to specify the new date(s). 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; *yy* is the last two digits of the year and is optional. For example:

settime 1008004583 *ralph pete*

sets the access and modification dates of files *ralph* and *pete* to Oct. 8, 12:45 AM, 1983. Another example:

settime -f *ralph john*

This sets the access and modification dates of the file, *john*, to those of the file, *ralph*.

NOTES

Use of touch in place of settime is encouraged.

NAME

tset - provides information for setting terminal modes.

SYNOPSIS

tset [*options*][*type*]

DESCRIPTION

tset allows the user to set a terminal's ERASE and KILL characters, and define the terminal's type and capabilities by creating values for the TERM and TERMCAP environment variables. If a *type* is given with the *-s* option, tset creates information for a terminal of the specified type. The type may be any type given in /usr/share/lib/termcap. If the *type* is not specified with the *-s* option, tset creates information for a terminal of the type defined by the value of the environment variable, TERM unless the *-h* or *-m* option is given. If the TERM variable is undefined, tset looks in /usr/share/lib/termcap for the appropriate information. If these options are used, tset searches the /etc/ttytype file for the terminal type corresponding to the current serial port; it then creates information for a terminal based on this type. If the serial port is not found in /etc/ttytype, the terminal type is set to dumb.

tset displays the created information at the standard output. The information is in a form that can be used to set the current environment variables. The exact form depends on the login shell from which tset was invoked. The examples below illustrate how to use this information to change the variables.

There are the following options

-e[cP]
: Sets the erase character to [*c*] on all terminals. The default setting is the BACKSPACE, or CTRL-H.

-E[c] Identical to the -e command except that it only operates on terminals that can BACKSPACE

-k[c] Sets the KILL character to *c*, defaulting to CTRL-U.

- Prints the terminal type on the standard output.

-s Outputs the "setenv" commands [for csh()], or "export" and assignment commands [for sh()]. The type of commands are determined by the user's login shell.

-S Only outputs the strings to be placed in the environment variables.

-r Prints the terminal type on the diagnostic output.

-Q Suppresses the printing of the "Erase set to" and "Kill set to" messages.

-I Suppresses printing of the terminal initialization strings.

tset is most useful when included in the .login [for csh()] or .profile [for sh()] file executed automatically at login, with -m option is given, the first correct mapping prevails.

Examples

```
tset gt42
   tset - mdialup>300:adm3a-mdialup:dw2-Qr-e#
   tset -mdial:ti733-mplug:?hp2621-munknown:?-e-k^U
```

To use the information created by the -s option for the Bourne shell, (sh), repeat these commands:

```
tset -s...>/tmp/tset$$
   /tmp/tset$$
   rm/tmp/tset$$
```

To use the information create for csh, use:

```
set noglob
   set term=('tset-S...') setenv TERM$term[1] setenv TERMCAP"$term[2]" unset
   term unset noglob
```

FILES

"/usr/share/lib/termcap" Terminal capability database.

SEE ALSO

stty(1), termcap(1), tty(1).

NAME

xinstall – installs commands

SYNOPSIS

xinstall [-c *dira*] [-f *dirb*] [-n *dirc*] [-o] [-a] *file* [*dirz* ...]

DESCRIPTION

xinstall is a command most commonly used in "makefiles" [see make(CP)] to xinstall 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 file. The program prints messages telling you exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirz* ...) are given, xinstall will search [using find(C)] a set of default directories (/usr/bin/usr/usr/bin, /etc, /usr/lib, and /usr/usr/lib, in that order) for a file with the same name as *file*. When the first occurrence is found, xinstall 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 (*dirz* ...) 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 in *dira*. Looks for *file* in *dira* and xinstalls it there if it is not found. If it is found, *xinstall* 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 exists, the mode and owner will be that of the existing file. May be used alone or with the -o or -s options.

-l Ignores default directory list, searching only through the given directories (*dirz* ...). 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.

-o If *file* is found, this option saves the "found" file by copying it to *oldfile* in the directory in which is was found. 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

find(1), make(1)

NAME

yes - Prints string repeatedly

SYNOPSIS

yes[*string*]

DESCRIPTION

yes repeatedly outputs "y", or if a single string argument is given, *arg* is output repeatedly. The command will continue indefinitely unless aborted. **yes** is useful in pipes to commands that prompt for input and require a "y" response for a yes. In this case, **yes** terminates when the command that it pipes to terminates so that no infinite loop occurs.

SYSTEM CALLS (2)

SYSTEM CALLS (2)

NAME

intro

Errnos

This section describes all the system calls. Many of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value which is almost always −1 or the NULL pointer. The individual descriptions specify the details. The following is a complete list of the error names and their descriptions.

EACCES	Search permission is denied for a component of the path prefix.
EDEADLK	A process' attempt to lock a file region would cause a deadlock between processes vying for control of that region.
EEXIST	The named file exists.
EFAULT	*buf* or *path* points to an invalid address.
EFAULT	*path* points outside the allocated address space of the process.
EINVAL	An invalid argument was specified mentioning an undefined signal in a call to the `signal` or `kill` routine. Also set by the functions described in the math package (3M).
EINTR	A signal was caught during the system call.
EISNAM	A XENIX name file (semaphore, shared data, etc.) was specified when not expected.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines.
ENAMETOOLONG	The length of the *path* argument exceeds {PATH_MAX}, or the length of a *path* component exceeds {NAME_MAX} while (_POSIX_NO_TRUNC) is in effect.
ENAVAIL	An opensem(2), `waitsem`(2) or `sigsem`(2) was issued to a XENIX semaphore that has not been initialized by a call to `creatsem`(2). A `sigsem` was issued to a XENIX semaphore out of sequence; i.e., before the process has issued the corresponding `waitsem` to the semaphore. An `nbwaitsem` was issued to a semaphore guarding a resource that is currently in use by another process. The semaphore that a process was waiting on has been left in an inconsistent state when the process controlling the semaphore exited without relinquishing control properly; i.e., without issuing a `waitsem` on the semaphore.

ENOENT	The named file does not exist or is the null pathname.
ENOENT	A component of the path prefix does not exist or is a null pathname.
ENOLCK	Cannot allocate a record lock for fcntl or locking.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
ENOSPC	No space is available.
ENOTDIR	A component of the path prefix is not a directory.
ENOTNAM	Not available. A `creatsem`, `opensem`(2), `waitsem`(2), or `sigsem`(2) was issued using and invalid XENIX semaphore identifier. Or, a process attempted a `sdget`(2) on a file that exists but is not shared data type.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*. does not exist or is a null pathname.
EPERM	The effective user ID of the process is not super-user.
EROFS	The directory in which the file is to be created is located on a read-only file system.

NAME

chsize – changes the size of a file

SYNOPSIS

```
cc [flag ...] file ... -lx
int chsize (int fildes, long size);
```

DESCRIPTION

fildes is a file descriptor obtained from a create, open, dup, fcntl, or pipe system call. chsize changes the size of the file associated with the file descriptor *fildes* to be exactly *size* bytes in length. The routine either truncates the file, or pads it with an appropriate number of bytes. If *size* is less than the initial size of the file, then all allocated disk blocks between *size* and the initial file size are freed.

The maximum file size as set by ulimit(2) is enforced when chsize is called, rather than on subsequent writes. Thus chsize fails, and the file size remains unchanged if the new changed file size would exceed the ulimit.

DIAGNOSTICS

Upon successful completion, a value of 0 is returned. Otherwise, the value -1 is returned and errno is set to indicate the error.

SEE ALSO

creat(2), dup(2), lseek(2), open(2), pipe(2), ulimit(2)

NOTES

In general if chsize is used to expand the size of a file, when data is written to the end of the file, intervening blocks are filled with zeros. In a some cases, reducing the file size may not remove the data beyond the new end-of-file.

NAME

creatsem – creates an instance of a binary semaphore.

SYNOPSIS

```
cc [flag ...] file ... -lx
int creatsem(int sem_num, int mode, char *sem_name);
```

Description

creatsem defines a binary semaphore named by *sem_name* to be used by waitsem and sigsem to manage mutually exclusive access to a resource, shared variable, or critical section of a program. creatsem returns a unique semaphore number, *sem_num*, which may then be used as the parameter in waitsem and sigsem calls. Semaphores are special files of 0 length. The filename space is used to provide unique identifiers for semaphores. *mode* sets the accessibility of the semaphore using the same format as file access bits. Access to a semaphore is granted only on the basis of the read access bit; the write and execute bits are ignored.

A semaphore can be operated on only by a synchronizing primitive, such as waitsem or sigsem, by creatsem which initializes it to some value, or by opensem which opens the semaphore for use by a process. Synchronizing primitives are guaranteed to be executed without interruption once started. These primitives are used by associating a semaphore with each resource (including critical code sections) to be protected.

The process controlling the semaphore should issue:

```
sem_num = creatsem("semaphore", mode);
```

to create, initialize, and open the semaphore for that process. All other processes using the semaphore should issue:

```
sem_num = opensem("semaphore");
```

to access the semaphore's identification value. Note that a process cannot open and use a semaphore that has not been initialized by a call to creatsem, nor should a process open a semaphore more than once in one period of execution. Both the creating and opening processes use waitsem and sigsem to use the semaphore *sem_num*.

DIAGNOSTICS

creatsem returns the value −1 if an error occurs. If the semaphore named by *sem_name* is already open for use by other processes, *errno* is set to EEXIST. If the file specified exists but is not a semaphore type, *errno* is set to ENOTNAM. If the semaphore has not been initialized by a call to creatsem, *errno* is set to EINVAL.

SEE ALSO

opensem(2), sigsem(2), waitsem(2).

NOTES

After a creatsem, you must do a waitsem to gain control of a given resource.

NAME

ftime – gets time and date

SYNOPSIS

```
cc [flag ...] file ... -lx [library ...]
#include <sys/times.h>

ftime(struct timeb *tp);
```

DESCRIPTION

ftime returns the time in a structure (see DIAGNOSTICS below). ftime will fail if tp points to an illegal address [EFAULT].

DIAGNOSTICS

The ftime entry fills in a structure pointed to by its argument, as defined by <sys/timeb.h>:

```
/* Structure returned by ftime system call */

struct timeb {
     long time;
     unsigned short millitm;
     short  timezone;
     short  dstflag;
};
```

Note that the timezone value is a system default timezone and not the value of the TZ environment variable.

The structure contains the time since the 00:00:00 GMT, January 1, 1970 up to 1000 milliseconds of more-precise interval, the local time zone (measured in minutes of time westward from Greenwich), and a flag that, if nonzero, indicates that Daylight Saving time applies locally during the appropriate part of the year.

SEE ALSO

stime(2), ctime(3C),

NOTES

Since ftime does not return the correct timezone value, its use is not recommended. See ctime(3C) for accurate use of the TZ variable.

NAME

lock – locks a process in primary memory.

SYNOPSIS

```
int lock(flag)
cc [flag ...] file ... -lx
```

DESCRIPTION

If the *flag* argument is nonzero, the process executing this call will not be swapped unless it is required to grow. If the argument is zero, the process is unlocked. This call may only be executed by the super-user. If someone other than the super-user tries to execute this call, a value of −1 is returned and the *errno* is set to EPERM.

NAME

locking – locks or unlocks a file region for reading or writing

SYNOPSIS

cc [flag ...] *file* ... -lx
locking (int *fildes*, int *mode*, long *size*)

DESCRIPTION

locking allows a specified number of bytes in a file to be controlled by the locking process. Other processes which attempt to read or write a portion of the file containing the locked region may sleep until the area become unlocked depending upon the mode in which the file region was locked.

A process that attempts to write to or read a file region that has been locked against reading and writing by another process (using the LK_LOCK or LK_NBLCK mode) with sleep until the region of the file has been released by the locking process.

A process that attempts to write to a file region that has been locked against writing by another process (using the LK_RLCK or LK_NBRLCK mode) will sleep until the region of the file has been released by the locking process, but a read request for that file region will proceed normally.

A process that attempts to lock a region of a file that contains areas that have been locked by other processes will sleep if it has specified the LK_LOCK or LK_RLCK mode in its lock request, but will return with the error EACCES if it specified LK_NBLCK or LK_NBRLCK.

fildes is the value returned from a successful create, open, dup, or pipe system call.

mode specifies the type of lock operation to be performed on the file region. The available values for mode are:

LK_UNLCK 0	Unlocks the specified region. The calling process releases a region of the file it has previously locked.
LK_LOCK 1	Locks the specified region. The calling process will sleep until the entire region is available if any part of it has been locked by a different process. The region is then locked for the calling process and no other process may read or write in any part of the locked region (lock against read and write).
LK_NBLCK 2	Locks the specified region. If any part of the region is already locked by a different process, return the error EACCES instead of waiting for the region to become available for locking (nonblocking lockrequest).
LK_RLCK 3	Same as LK_LOCK except that the locked region may be read by other processes (read permitted lock).
LK_NBRLCK 4	Same as LK_NBLCK except that the locked region may be read by other processes (nonblocking, read permitted lock).

The locking utility uses the current file pointer position as the starting point for the locking of the file segment. So a typical sequence of commands to lock a specific range within a file might be as follows:

```
fd=open("datafile",O_RDWR);
lseek(fd, 200L, 0);
locking(fd, LK_LOCK, 200L);
```

Accordingly, to lock or unlock an entire file a seek to the beginning of the file (position 0) must be done and then a locking call must be executed with a size of 0.

size is the number of contiguous bytes to be locked for unlocked. The region to be locked starts at the current offset in the file. If size is 0, the entire file is locked or unlocked. size may extend beyond the end of the file, in which case only the process issuing the lock call may access or add information to the file within the boundary defined by size.

The potential for a deadlock occurs when a process controlling a locked area is put to sleep by accessing another process' locked area. Thus calls to locking, read, or write scan for a deadlock prior to sleeping on a locked region. An EDEADLK error return is made if sleeping on the locked region would cause a deadlock.

Lock requests may, in whole or part, contain or be contained by a previously locked region for the same process. When this occurs, or when adjacent regions are locked, the regions are combined into a single area if the mode of the lock is the same (i.e.; either read permitted or regular lock). If the mode of the overlapping locks differ, the locked areas will be assigned assuming that the most recent request must be satisfied. Thus if a read only lock is applied to a region, or part of a region, that had been previously locked by the same process against both reading and writing, the area of the file specified by the new lock will be locked for read only, while the remaining region, if any, will remain locked against reading and writing. There is no arbitrary limit to the number of regions which may be locked in a file.

Unlock requests may, in whole or part, release one or more locked regions controlled by the process. When regions are not fully released, the remaining areas are still locked by the process. Release of the center section of a locked area requires an additional locked element to hold the separated section. If the lock table is full, an error is returned, and the requested region is not released. Only the process which locked the file region may unlock it. An unlock request for a region that the process does not have locked, or that is already unlocked, has no effect. When a process terminates, all locked regions controlled by that process are unlocked.

If a process has done more than one open on a file, all locks put on the file by that process will be released on the first close of the file.

Although no error is returned if locks are applied to special files or pipes, read/write operations on these types of files will ignore the locks. Locks may not be applied to a directory.

SEE ALSO

close(2) creat(2), dup(2), lseek(2), open(2), read(2), write(2)

DIAGNOSTICS

locking returns the value (int)-1 if an error occurs. If any portion of the region has been locked by another process for the LK_LOCK and LK_RLCK actions and the lock request is to test only, errno is set to EAGAIN. If locking the region would cause a deadlock, errno is set to EDEADLK If an internal lock cannot be allocated, errno is set to ENOLCK.

NAME

mknod – make a directory, or a special or ordinary file

SYNOPSIS

```
#include <sys/types.h>
#include <osfcn.h>
#include <sys/stat.h>

int mknod (const char *path, mode_t mode, dev_t dev);
```

DESCRIPTION

mknod creates a new file named by the path name pointed to by *path*. The file type and permissions of the new file are initialized from *mode*.

The file type is specified in *mode* by the S_IFMT bits, which must be set to one of the following values:

S_IFIFO	fifo special
S_IFCHR	character special
S_IFDIR	directory
S_IFBLK	block special
S_IFREG	ordinary file
S_IFNAM	name special file

The file access permissions are specified in *mode* by the 0007777 bits, and may be constructed by an OR of the following values:

S_ISUID	04000	Set user ID on execution.
S_ISGID	020#0	Set group ID on execution if # is 7, 5, 3, or 1 Enable mandatory file/record locking if # is 6, 4, 2, or 0
S_ISVTX	01000	Save text image after execution.
S_IRUSR	00400	Read by owner.
S_IWUSR	00200	Write by owner.
S_IXUSR	00100	Execute (search if a directory) by owner.
S_IRWXG	00070	Read, write, execute by group.
S_IRGRP	00040	Read by group.
S_IWGRP	00020	Write by group.
S_IXGRP	00010	Execute by group.
S_IRWXO	00007	Read, write, execute (search) by others.
S_IROTH	00004	Read by others.
S_IWOTH	00002	Write by others
S_IXOTH	00001	Execute by others.

The owner ID of the file is set to the effective user ID of the process. The group ID of the file is set to the effective group ID of the process. However, if the S_ISGID bit is set in the parent directory, then the group ID of the file is inherited from the parent. If the group ID of the new file does not match the effective group ID or one of the supplementary group IDs, the S_ISGID bit is cleared.

Values of *mode* other than those above are undefined and should not be used. The access permission bits of *mode* are modified by the process's file mode creation mask: all bits set in the process's file mode creation mask are cleared [see umask(2)]. For block and character special files, *dev* is the special file's device number. For name special files, *dev* is the file type of the name file, either a

XENIX shared data file or a XENIX semaphore. Otherwise, *dev* is ignored. See mkdev(3C).

mknod may be invoked only by the privileged user for file types other than FIFO special.

mknod fails and creates no new file if one or more of the following are true:

EEXIST	The named file exists.
EINVAL	Invalid *arg* value.
EFAULT	*path* points outside the allocated address space of the process.
ELOOP	Too many symbolic links were encountered in translating *path*.
EMULTIHOP	Components of *path* require hopping to multiple remote machines.
ENAMETOOLONG	The length of the *path* argument exceeds {PATH_MAX}, or the length of a *path* component exceeds {NAME_MAX} while (_POSIX_NO_TRUNC) is in effect.
ENOTDIR	A component of the path prefix is not a directory.
ENOENT	A component of the path prefix does not exist or is a null pathname.
EPERM	The effective user ID of the process is not super-user.
EROFS	The directory in which the file is to be created is located on a read-only file system.
ENOSPC	No space is available.
EINTR	A signal was caught during the mknod system call.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.

SEE ALSO

chmod(2), exec(2), umask(2), mkdev(3C), mkfifo(3C), stat(5).
fs(4) in the *System Administrator's Reference Manual.*
mkdir(1) in the *User's Reference Manual.*
creatsem(2), sdget(2) in the *BSD/XENIX Compatibility Guide.*

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and *errno* is set to indicate the error.

NOTES

If mknod creates a device in a remote directory using Remote File Sharing, the major and minor device numbers are interpreted by the server.

Semaphore files should be created with the creatsem system call. Shared data files should be created with the sdget system call.

NAME

nap – suspends execution for a short interval

SYNOPSIS

```
cc [flag ...] file ... -lx
long nap (long period);
```

DESCRIPTION

The current process is suspended from execution for at least the number of milliseconds specified by *period*, or until a signal is received.

DIAGNOSTICS

On successful completion, a long integer indicating the number of milliseconds actually slept is returned. If the process received a signal while napping, the return value will be -1, and errno will be set to EINTR.

SEE ALSO

sleep(2)

NOTES

This function is driven by the system clock, which in most cases has a granularity of tens of milliseconds.

NAME

opensem – opens a semaphore

SYNOPSIS

```
cc [flag ...] file ... -lx
int opensem(int sem_name, char *sem_num);
```

DESCRIPTION

opensem opens a semaphore named by *sem_name* and returns the unique semaphore identification number *sem_num* used by `waitsem` and `sigsem`. `creatsem` should always be called to initialize the semaphore before the first attempt to open it.

DIAGNOSTICS

opensem returns a value of −1 if an error occurs. If the semaphore named does not exist, *errno* is set to `ENOENT`. If the file specified is not a semaphore file (i.e., a file previously created by a process using a call to `creatsem`), *errno* is set to `ENOTNAM`. If the semaphore has become invalid due to inappropriate use, *errno is set to* `ENAVAIL`.

SEE ALSO

`creatsem`(2), `sigsem`(2), `waitsem`(2).

WARNING

It is not advisable to open the same semaphore more than once. Although it is possible to do this, it may result in a serious deadlock.

NAME

rdchk – checks to see if there is data to be read

SYNOPSIS

```
cc [flag ...] file ... -lx
rdchk(int fdes);
```

DESCRIPTION

rdchk checks to see if a process will block if it attempts to read the file designated by fdes. rdchk returns 1 if there is data to be read or if it is the end of the file (EOF). In this context, the proper sequence of calls using rdchk is:

```
if(rdchk(fildes) > 0)
    read(fildes, buffer, nbytes);
```

DIAGNOSTICS

rdchk returns -1 if an error occurs (e.g., EBADF), 0 if the process will block if it issues a read and 1 if it is okay to read. EBADF is returned if a rdchk is done on a semaphore file or if the file specified doesn't exist.

SEE ALSO

read(2)

NAME

sdenter, sdleave – synchronizes access to a shared data segment.

SYNOPSIS

```
cc [flag ...] file ... -lx
#include <sys/sd.h>

int sdenter(char *addr, int flags); int sdleave(char *addr);
```

DESCRIPTION

sdenter is used to indicate that the current process is about to access the contents of a shared data segment. The actions performed depend on the value of *flags*. *flags* values are formed by OR-ing together entries from the following list: If another process has called sdenter but not sdleave for the indicated segment, and the segment was not created with the SD_UNLOCK flag set, return an ENAVAIL error instead of waiting for the segment to become free. Indicates that the process wants to write data to the shared data segment. A process that has attached to a shared data segment with the SD_RDONLY flag set will not be allowed to enter with the SD_WRITE flag set. sdleave is used to indicate that the current process is done modifying the contents of a shared data segment.

Only changes made between invocations of sdenter and sdleave are guaranteed to be reflected in other processes. sdenter and sdleave are very fast; consequently, it is recommended that they be called frequently rather than leave sdenter in effect for any period of time. In particular, system calls should be avoided between sdenter and sdleave calls.

The fork system call is forbidden between calls to sdenter and sdleave if the segment was created without the SD_UNLOCK flag.

DIAGNOSTICS

Successful calls return 0. Unsuccessful calls return −1 and *errno* is set to indicate the error. *errno* is set to EINVAL if a process does an sdenter with the SD_WRITE flag set and the segment is already attached with the SD_RDONLY flag set. *errno* is set to ENAVAIL if the SD_NOWAIT flag is set for *sdenter* and the shared data segment is not free.

SEE ALSO

sdget(2), sdgetv(2).

NAME

sdget, sdfree - attaches and detaches a shared data segment.

SYNOPSIS

```
cc [flag ...] file ... -lx
#include <sys/sd.h>

char *sdget(char *path, in flags, /* long size, int mode */;

int sdfree(char *addr);
```

DESCRIPTION

sdget attaches a shared data segment to the data space of the current process. The actions performed are controlled by the value of *flags*. *flags* values are constructed by OR-ing flags from the following list: Attach the segment for reading only. Attach the segment for both reading and writing. If the segment named by *path* exists and is not in use (active), this flag will have the same effect as creating a segment from scratch. Otherwise, the segment is created according to the values of *size* and *mode*. Read and write access to the segment is granted to other processes based on the permissions passed in *mode*, and functions the same as those for regular files. Execute permission is meaningless. The segment is initialized to contain all zeroes. If the segment is created because of this call, the segment will be made so that more than one process can be between sdenter and sdleave calls.

sdfree detaches the current process from the shared data segment that is attached at the specified address. If the current process has done sdenter but not an sdleave for the specified segment, sdleave will be done before detaching the segment.

When no process remains attached to the segment, the contents of that segment disappear, and no process can attach to the segment without creating it by using the SD_CREAT flag in sdget. *errno* is set to EEXIST if a process tries to create a shared data segment that exists and is in use. *errno* is set to ENOTNAM if a process attempts an sdget on a file that exists but is not a shared data type.

DIAGNOSTICS

On successful completion, the address at which the segment was attached is returned. Otherwise, −1 is returned, and *errno* is set to indicate the error. *errno* is set to EINVAL if a process does an sdget on a shared data segment to which it is already attached. *errno* is set to EEXIST if a process tries to create a shared data segment that exists an is in use. *errno* is set to ENOTNAM if a process attempts an sdget on a file that exists but is not a shared data type.

The mode parameter must be included on the first call of the sdget function.

SEE ALSO

sdenter(2), sdgetv(2).

NAME

sdgetv - synchronizes shared data access.

SYNOPSIS

```
cc [flag ...] file ... -lx
#include <sys/sd.h>

int sdgetv(addr)
int sdwaitv(char *addr, int vnum);
```

DESCRIPTION

sdgetv and sdwaitv may be used to synchronize cooperating processes that are using shared data segments. The return value of both routines is the version number of the shared data segment attached to the process at address *addr*. The version number of a segment changes whenever some process does an sdleave for that segment.

sdgetv simply returns the version number of the indicated segment.

sdwaitv forces the current process to sleep until the version number for the indicated segment is no longer equal to *vnum*.

DIAGNOSTICS

Upon successful completion, both sdgetv and sdwaitv return a positive integer that is the current version number for the indicated shared data segment. Otherwise, a value of −1 is returned, and *errno* is set to indicate the error.

SEE ALSO

sdenter(2), sdget(2).

NAME

sigsem - signals a process waiting on a semaphore.

SYNOPSIS

```
cc [flag ...] file ... -lx
sigsem(int sem_num);
```

DESCRIPTION

sigsem signals a process that is waiting on the semaphore *sem_num* that it may proceed and use the resource governed by the semaphore. sigsem is used in conjunction with waitsem to allow synchronization of processes wishing to access a resource. One or more processes may waitsem on the given semaphore and will be put to sleep until the process which currently has access to the resource issues a sigsem call. If there are any waiting processes, sigsem causes the process which is next in line on the semaphore's queue to be rescheduled for execution. The semaphore's queue is organized in First In, First Out (FIFO) order.

DIAGNOSTICS

sigsem returns the value (int) −1 if an error occurs. If *sem_num* does not refer to a semaphore type file, *errno* is set to ENOTNAM. If *sem_num* has not been previously opened by opensem, *errno* is set to EBADF. If the process issuing a sigsem call is not the current "owner" of the semaphore (i.e., if the process has not issued a waitsem call before the sigsem), *errno* is set to ENAVAIL.

SEE ALSO

creatsem(2), opensem(2), waitsem(2).

NAME

stat, lstat, fstat – get file status

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>

int stat (const char *path, struct stat *buf);

int lstat (const char *path, struct stat *buf);

int fstat (int fildes, struct stat *buf);
```

DESCRIPTION

path points to a path name naming a file. Read, write, or execute permission of the named file is not required, but all directories listed in the path name leading to the file must be searchable. `stat` obtains information about the named file.

Note that in a Remote File Sharing environment, the information returned by `stat` depends on the user/group mapping set up between the local and remote computers. [See `idload`(1M).]

`lstat` obtains file attributes similar to `stat`, except when the named file is a symbolic link; in that case `lstat` returns information about the link, while `stat` returns information about the file the link references.

`fstat` obtains information about an open file known by the file descriptor *fildes*, obtained from a successful `open`, `creat`, `dup`, `fcntl`, or `pipe` system call.

buf is a pointer to a `stat` structure into which information is placed concerning the file.

The contents of the structure pointed to by *buf* include the following members:

```
mode_t  st_mode;   /* File mode [see mknod(2)] */
ino_t   st_ino;    /* Inode number */
dev_t   st_dev;    /* ID of device containing */
                   /* a directory entry for this file */
dev_t   st_rdev;   /* ID of device */
                   /* This entry is defined only for */
                   /* character special files */,
                   /* XENIX special named files or block
                   /* special files */
nlink_t st_nlink;  /* Number of links */
uid_t   st_uid;    /* User ID of the file's owner */
gid_t   st_gid;    /* Group ID of the file's group */
off_t   st_size;   /* File size in bytes */
time_t  st_atime;  /* Time of last access */
time_t  st_mtime;  /* Time of last data modification */
time_t  st_ctime;  /* Time of last file status change */
                   /* Times measured in seconds since */
                   /* 00:00:00 GMT, Jan. 1, 1970 */
```

st_mode The mode of the file as described in mknod(2).

st_ino This field uniquely identifies the file in a given file system. The pair st_ino and st_dev uniquely identifies regular files.

st_dev This field uniquely identifies the file system that contains the file. Its value may be used as input to the ustat system call to determine more information about this file system. No other meaning is associated with this value.

st_rdev This field should be used only by administrative commands. It is valid only for block special files or character special files or XENIX special named files. The *st_rdev* field for block special and character special files only has meaning on the system where the file was configured.

If the file is a XENIX special named file, it contains the type code [see stat(4) for the XENIX semaphore and shared data type code values S_INSEM and S_INSHD].

st_nlink This field should be used only by administrative commands.

st_uid The user ID of the file's owner.

st_gid The group ID of the file's group.

st_size For regular files, this is the address of the end of the file. For pipes or FIFOs, this is the count of the data currently in the file. For block special character special, or XENIX special named files. this is not defined.

st_atime Time when file data was last accessed. Changed by the following system calls: creat, mknod, pipe, utime, read, creatsem, opensem, sigsem, waitsem, sdget and sdfree.

st_mtime Time when data was last modified. Changed by the following system calls: creat, mknod, pipe, utime, write.

st_ctime Time when file status was last changed. Changed by the following system calls: chmod, chown, creat, link, mknod, pipe, unlink, utime, write, creatsem, sdget and sdfree.

stat and lstat fail if one or more of the following are true:

EACCES Search permission is denied for a component of the path prefix.

EBADF *fildes* is not a valid open file descriptor.

EFAULT *buf* or *path* points to an invalid address.

EINTR A signal was caught during the stat system call.

ELOOP Too many symbolic links were encountered in translating *path*.

EMULTIHOP	Components of *path* require hopping to multiple remote machines.
ENAMETOOLONG	The length of the *path* argument exceeds {PATH_MAX}, or the length of a *path* component exceeds {NAME_MAX} while (_POSIX_NO_TRUNC) is in effect.
ENOENT	The named file does not exist or is the null pathname.
ENOTDIR	A component of the path prefix is not a directory.
ENOLINK	*path* points to a remote machine and the link to that machine is no longer active.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*.

`fstat` fails if one or more of the following are true:

ENOLINK	*fildes* points to a remote machine and the link to that machine is no longer active.
EOVERFLOW	A component is too large to store in the structure pointed to by *buf*.

SEE ALSO

chmod(2), chown(2), creat(2), link(2), mknod(2), pipe(2), read(2), time(2), unlink(2), utime(2), write(2), stat(5).

DIAGNOSTICS

Upon successful completion a value of 0 is returned. Otherwise, a value of −1 is returned and *errno* is set to indicate the error.

NAME

waitsem, nbwaitsem - awaits and checks access to a resource governed by a semaphore.

SYNOPSIS

```
cc [flag ...] file ... -lx
waitsem(int sem_num);

nbwaitsem(int sem_num);
```

DESCRIPTION

waitsem gives the calling process access to the resource governed by the semaphore *sem_num*. If the resource is in use by another process, waitsem will put the process to sleep until the resource becomes available; nbwaitsem will return the error ENAVAIL. waitsem and nbwaitsem are used in conjunction with sigsem to allow synchronization of processes withing to access a resource. One or more processes may waitsem on the given semaphore and will be put to sleep until the process which currently has access to the resource issues sigsem. sigsem causes the process which is next in line on the semaphore's queue to be rescheduled for execution. The semaphore's queue is organized in First In, First Out (FIFO) order.

DIAGNOSTICS

waitsem returns the value (int) −1 if an error occurs. If *sem_num* has not been previously opened by a call to opensem or creatsem, *errno* is set to EBADF. If *sem_num* does not refer to a semaphore type file, *errno* is set to ENOTNAM. All processes waiting (or attempting to wait) on the semaphore return with *errno* set to ENAVAIL when the process controlling the semaphore exits without relinquishing control (thereby leaving the resource in an undeterminate state). If a process does two waitsems in a row without doing a intervening sigsem, *errno* is set to EINVAL.

SEE ALSO

opensem(2), creatsem(2).

FILE FORMATS (4)

FILE FORMATS (4)

NAME

stat – data returned by stat system call

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
```

DESCRIPTION

The system calls stat, lstat and fstat return data in a stat structure, which is defined in stat.h:

```
struct   stat
{
        dev_t     st_dev;
        ino_t     st_ino;
        mode_t    st_mode;
        nlink_t   st_nlink;
        uid_t     st_uid;
        gid_t     st_gid;
        dev_t     st_rdev;
        off_t     st_size;
        time_t    st_atime;
        time_t    st_mtime;
        time_t    st_ctime;
};
```

The constants used in the st_mode field are also defined in this file:

```
#define S_IFMT    0xF000    /* type of file */
#define S_IAMB    0x1FF     /* access mode bits */
#define S_IFIFO   0x1000    /* fifo */
#define S_IFCHR   0x2000    /* character special */
#define S_IFDIR   0x4000    /* directory */
#define S_IFNAM   0x5000    /* XENIX special named file */
#define S_INSEM   0x1       /* XENIX semaphore subtype of IFNAM */
#define S_INSEM   0x2       /* XENIX shared data subtype of IFNAM */
#define S_IFBLK   0x6000    /* block special */
#define S_IFREG   0x8000    /* regular */
#define S_IFLNK   0xA000    /* symbolic link */
#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_IREAD   00400     /* read permission, owner */
#define S_IWRITE  00200     /* write permission, owner */
#define S_IEXEC   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 */
#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

stat(2), types(5).

XENIX Subject Index

307-324

UNIX
PRESS
A Prentice Hall Title