-

UNIX PROGRAMMER'S MANUAL

Sixth Edition

K. Thompson

D. M. Ritchie

May, 1975

-

This manual was set by a Graphic Systems phototypesetter driven by the *troff* formatting program operating under the UNIX system. The text of the manual was prepared using the *ed* text editor.

-

PREFACE
to the Sixth Edition

We are grateful to L. L. Cherry, R. C. Haight, S. C. Johnson, B. W. Kernighan, M. E. Lesk, and E. N. Pinson for their contributions to the system software, and to L. E. McMahon for software and for his contributions to this manual. We are particularly appreciative of the invaluable technical, editorial, and administrative efforts of J. F. Ossanna, M. D. McIlroy, and R. Morris. They all contributed greatly to the stock of UNIX software and to this manual. Their inventiveness, thoughtful criticism, and ungrudging support increased immeasurably not only whatever success the UNIX system enjoys, but also our own enjoyment in its creation.

-

INTRODUCTION TO THIS MANUAL

This manual gives descriptions of the publicly available features of UNIX. It provides neither a general overview – see "The UNIX Time-sharing System" (Comm. ACM **17** 7, July 1974, pp. 365-375) for that – nor details of the implementation of the system, which remain to be disclosed.

Within the area it surveys, this manual attempts to be as complete and timely as possible. A conscious decision was made to describe each program in exactly the state it was in at the time its manual section was prepared. In particular, the desire to describe something as it should be, not as it is, was resisted. Inevitably, this means that many sections will soon be out of date.

This manual is divided into eight sections:

I. Commands
II. System calls
III. Subroutines
IV. Special files
V. File formats and conventions
VI. User-maintained programs
VII. User-maintained subroutines
VIII. Maintenance

Commands are programs intended to be invoked directly by the user, in contradistinction to subroutines, which are intended to be called by the user's programs. Commands generally reside in directory */bin* (for *bin*ary programs). Some programs also reside in */usr/bin,* to save space in */bin.* These directories are searched automatically by the command interpreter.

System calls are entries into the UNIX supervisor. In assembly language, they are coded with the use of the opcode *sys*, a synonym for the *trap* instruction. In this edition, the C language interface routines to the system calls have been incorporated in section II.

A small assortment of subroutines is available; they are described in section III. The binary form of most of them is kept in the system library */lib/liba.a.* The subroutines available from C and from Fortran are also included; they reside in */lib/libc.a* and */lib/libf.a* respectively.

The special files section IV discusses the characteristics of each system "file" which actually refers to an I/O device. The names in this section refer to the DEC device names for the hardware, instead of the names of the special files themselves.

The file formats and conventions section V documents the structure of particular kinds of files; for example, the form of the output of the loader and assembler is given. Excluded are files used by only one command, for example the assembler's intermediate files.

User-maintained programs and subroutines (sections VI and VII) are not considered part of the UNIX system, and the principal reason for listing them is to indicate their existence without necessarily giving a complete description. The authors of the individual programs should be consulted for more information.

Section VIII discusses commands which are not intended for use by the ordinary user, in some cases because they disclose information in which he is presumably not interested, and in others because they perform privileged functions.

Each section consists of a number of independent entries of a page or so each. The name of the entry is in the upper corners of its pages, its preparation date in the upper middle. Entries within each section are alphabetized. The page numbers of each entry start at 1. (The earlier hope for frequent, partial updates of the manual is clearly in vain, but in any event it is not feasible to maintain consecutive page numbering in a document like this.)

-

All entries are based on a common format, not all of whose subsections will always appear.

The *name* section repeats the entry name and gives a very short description of its purpose.

The *synopsis* summarizes the use of the program being described. A few conventions are used, particularly in the Commands section:

Boldface words are considered literals, and are typed just as they appear.

Square brackets ([ ]) around an argument indicate that the argument is optional. When an argument is given as ''name'', it always refers to a file name.

Ellipses ''. . .'' are used to show that the previous argument-prototype may be repeated.

A final convention is used by the commands themselves. An argument beginning with a minus sign ''_'' is often taken to mean some sort of flag argument even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with ''_''.

The *description* section discusses in detail the subject at hand.

The *files* section gives the names of files which are built into the program.

A *see also* section gives pointers to related information.

A *diagnostics* section discusses the diagnostic indications which may be produced. Messages which are intended to be self-explanatory are not listed.

The *bugs* section gives known bugs and sometimes deficiencies. Occasionally also the suggested fix is described.

At the beginning of this document is a table of contents, organized by section and alphabetically within each section. There is also a permuted index derived from the table of contents. Within each index entry, the title of the writeup to which it refers is followed by the appropriate section number in parentheses. This fact is important because there is considerable name duplication among the sections, arising principally from commands which exist only to exercise a particular system call.

This manual was prepared using the UNIX text editor *ed* and the formatting program *troff*.

-

HOW TO GET STARTED

This section provides the basic information you need to get started on UNIX: how to log in and log out, how to communicate through your terminal, and how to run a program. See ''UNIX for Beginners'' by Brian W. Kernighan for a more complete introduction to the system.

Logging in. You must call UNIX from an appropriate terminal. UNIX supports ASCII terminals typified by the TTY 37, the GE Terminet 300, the Dasi 300, and various graphical terminals. You must also have a valid user name, which may be obtained, together with the telephone number, from the system administrators. The same telephone number serves terminals operating at all the standard speeds. After a data connection is established, the login procedure depends on what kind of terminal you are using.

> *300-baud terminals:* Such terminals include the GE Terminet 300, most display terminals, Execuport, TI, GSI, and certain Anderson-Jacobson terminals. These terminals generally have a speed switch which should be set at ''300'' (or ''30'' for 30 characters per second) and a half/full duplex switch which should be set at full-duplex. (This switch will often have to be changed since many other systems require half-duplex). When a connection is established, the system types ''login:''; you type your user name, followed by the ''return'' key. If you have a password, the system asks for it and turns off the printer on the terminal so the password will not appear. After you have logged in, the ''return'', ''new line'', or ''linefeed'' keys will give exactly the same results.
>
> *TTY 37 terminal:* When you have established a data connection, the system types out a few garbage characters (the ''login:'' message at the wrong speed). Depress the ''break'' (or ''interrupt'') key; this is a speed-independent signal to UNIX that a 150-baud terminal is in use. The system then will type ''login:,'' this time at the correct speed; you respond with your user name. From the TTY 37 terminal, and any other which has the ''new-line'' function (combined carriage return and linefeed), terminate each line you type with the ''new-line'' key (*not* the ''return'' key).

For all these terminals, it is important that you type your name in lower-case if possible; if you type upper-case letters, UNIX will assume that your terminal cannot generate lower-case letters and will translate all subsequent upper-case letters to lower case.

The evidence that you have successfully logged in is that the Shell program will type a ''%'' to you. (The Shell is described below under ''How to run a program.'')

For more information, consult *getty* (VIII), which discusses the login sequence in more detail, and *tty* (IV), which discusses typewriter I/O.

Logging out. There are three ways to log out:

> You can simply hang up the phone.
>
> You can log out by typing an end-of-file indication (EOT character, control ''d'') to the Shell. The Shell will terminate and the ''login: '' message will appear again.
>
> You can also log in directly as another user by giving a *login* command (I).

How to communicate through your terminal. When you type to UNIX, a gnome deep in the system is gathering your characters and saving them in a secret place. The characters will not be given to a program until you type a return (or new-line), as described above in *Logging in.*

UNIX typewriter I/O is full-duplex. It has full read-ahead, which means that you can type at any time, even while a program is typing at you. Of course, if you type during output, the output will have the input characters interspersed. However, whatever you type will be saved up and interpreted in correct sequence. There is a limit to the amount of read-ahead, but it is generous and not likely to be exceeded unless the system is in trouble. When the read-ahead limit is exceeded, the system throws away all the saved characters.

On a typewriter input line, the character ''@'' kills all the characters typed before it, so typing mistakes can be repaired on a single line. Also, the character ''#'' erases the last character typed. Successive uses of

-

"#" erase characters back to, but not beyond, the beginning of the line. "@" and "#" can be transmitted to a program by preceding them with "\". (So, to erase "\", you need two "#"s).

The ASCII "delete" (a.k.a. "rubout") character is not passed to programs but instead generates an *interrupt signal.* This signal generally causes whatever program you are running to terminate. It is typically used to stop a long printout that you don't want. However, programs can arrange either to ignore this signal altogether, or to be notified when it happens (instead of being terminated). The editor, for example, catches interrupts and stops what it is doing, instead of terminating, so that an interrupt can be used to halt an editor printout without losing the file being edited.

The *quit* signal is generated by typing the ASCII FS character. It not only causes a running program to terminate but also generates a file with the core image of the terminated process. Quit is useful for debugging.

Besides adapting to the speed of the terminal, UNIX tries to be intelligent about whether you have a terminal with the new-line function or whether it must be simulated with carriage-return and line-feed. In the latter case, all input carriage returns are turned to new-line characters (the standard line delimiter) and both a carriage return and a line feed are echoed to the terminal. If you get into the wrong mode, the *stty* command (I) will rescue you.

Tab characters are used freely in UNIX source programs. If your terminal does not have the tab function, you can arrange to have them turned into spaces during output, and echoed as spaces during input. The system assumes that tabs are set every eight columns. Again, the *stty* command (I) will set or reset this mode. Also, there is a file which, if printed on TTY 37 or TermiNet 300 terminals, will set the tab stops correctly (*tabs* (V)).

Section *tty* (IV) discusses typewriter I/O more fully.

How to run a program; the Shell. When you have successfully logged into UNIX, a program called the Shell is listening to your terminal. The Shell reads typed-in lines, splits them up into a command name and arguments, and executes the command. A command is simply an executable program. The Shell looks first in your current directory (see next section) for a program with the given name, and if none is there, then in a system directory. There is nothing special about system-provided commands except that they are kept in a directory where the Shell can find them.

The command name is always the first word on an input line; it and its arguments are separated from one another by spaces.

When a program terminates, the Shell will ordinarily regain control and type a "%" at you to indicate that it is ready for another command.

The Shell has many other capabilities, which are described in detail in section *sh* (I).

The current directory. UNIX has a file system arranged in a hierarchy of directories. When the system administrator gave you a user name, he also created a directory for you (ordinarily with the same name as your user name). When you log in, any file name you type is by default in this directory. Since you are the owner of this directory, you have full permissions to read, write, alter, or destroy its contents. Permissions to have your will with other directories and files will have been granted or denied to you by their owners. As a matter of observed fact, few UNIX users protect their files from destruction, let alone perusal, by other users.

To change the current directory (but not the set of permissions you were endowed with at login) use *chdir* (I).

Path names. To refer to files not in the current directory, you must use a path name. Full path names begin with "/", the name of the root directory of the whole file system. After the slash comes the name of each directory containing the next sub-directory (followed by a "/") until finally the file name is reached. E.g.: */usr/lem/filex* refers to the file *filex* in the directory *lem; lem* is itself a subdirectory of *usr; usr* springs directly from the root directory.

If your current directory has subdirectories, the path names of files therein begin with the name of the sub-

-

directory (no prefixed "/").

Without important exception, a path name may be used anywhere a file name is required.

Important commands which modify the contents of files are *cp* (I), *mv* (I), and *rm* (I), which respectively copy, move (i.e. rename) and remove files. To find out the status of files or directories, use *ls* (I). See *mkdir* (I) for making directories; *rmdir* (I) for destroying them.

For a fuller discussion of the file system, see "The UNIX Time-Sharing System," by the present authors. It may also be useful to glance through section II of this manual, which discusses system calls, even if you don't intend to deal with the system at that level.

Writing a program. To enter the text of a source program into a UNIX file, use *ed* (I). The three principal languages in UNIX are assembly language (see *as* (I)), Fortran (see *fc* (I)), and C (see *cc* (I)). After the program text has been entered through the editor and written on a file, you can give the file to the appropriate language processor as an argument. The output of the language processor will be left on a file in the current directory named "a.out". (If the output is precious, use *mv* to move it to a less exposed name soon.) If you wrote in assembly language, you will probably need to load the program with library subroutines; see *ld* (I). The other two language processors call the loader automatically.

When you have finally gone through this entire process without provoking any diagnostics, the resulting program can be run by giving its name to the Shell in response to the "%" prompt.

Next, you will need *cdb* (I) or *db* (I) to examine the remains of your program. The former is useful for C programs, the latter for assembly-language. No debugger is much help for Fortran.

Your programs can receive arguments from the command line just as system programs do. See *exec* (II).

Text processing. Almost all text is entered through the editor. The commands most often used to write text on a terminal are: *cat, pr, roff, nroff,* and *troff,* all in section I.

The *cat* command simply dumps ASCII text on the terminal, with no processing at all. The *pr* command paginates the text, supplies headings, and has a facility for multi-column output. *Troff* and *nroff* are elaborate text formatting programs, and require careful forethought in entering both the text and the formatting commands into the input file. *Troff* drives a Graphic Systems phototypesetter; it was used to produce this manual. *Nroff* produces output on a typewriter terminal. *Roff* (I) is a somewhat less elaborate text formatting program, and requires somewhat less forethought.

Surprises. Certain commands provide inter-user communication. Even if you do not plan to use them, it would be well to learn something about them, because someone else may aim them at you.

To communicate with another user currently logged in, *write* (I) is used; *mail* (I) will leave a message whose presence will be announced to another user when he next logs in. The write-ups in the manual also suggest how to respond to the two commands if you are a target.

When you log in, a message-of-the-day may greet you before the first "%".

-

TABLE OF CONTENTS

I. COMMANDS

ar . . . . . archive and library maintainer
as . . . . . assembler
bas . . . . . basic
bc . . . . . arbitrary precision interactive language
cat . . . . . concatenate and print
cc . . . . . C compiler
cdb . . . . . C debugger
chdir . . . . . change working directory
chmod . . . . . change mode
cmp . . . . . compare two files
comm . . . . . print lines common to two files
cp . . . . . copy
cref . . . . . make cross reference listing
date . . . . . print and set the date
db . . . . . debug
dc . . . . . desk calculator
dd . . . . . convert and copy a file
diff . . . . . differential file comparator
dsw . . . . . delete interactively
du . . . . . summarize disk usage
echo . . . . . echo arguments
ed . . . . . text editor
eqn . . . . . typeset mathematics
exit . . . . . terminate command file
fc . . . . . Fortran compiler
file . . . . . determine file type
find . . . . . find files
goto . . . . . command transfer
grep . . . . . search a file for a pattern
if . . . . . conditional command
kill . . . . . terminate a process
ld . . . . . link editor
ln . . . . . make a link
login . . . . . sign onto UNIX
ls . . . . . list contents of directory
mail . . . . . send mail to designated users
man . . . . . run off section of UNIX manual
mesg . . . . . permit or deny messages
mkdir . . . . . make a directory
mv . . . . . move or rename a file
neqn . . . . . typeset mathematics on terminal
newgrp . . . . . log in to a new group
nice . . . . . run a command at low priority
nm . . . . . print name list
nohup . . . . . run a command immune to hangups
nroff . . . . . format text
od . . . . . octal dump
opr . . . . . off line print
passwd . . . . . change login password

pfe . . . . . print floating exception
pr . . . . . print file
prof . . . . . display profile data
ps . . . . . process status
pwd . . . . . working directory name
rc . . . . . Ratfor compiler
rev . . . . . reverse lines of a file
rm . . . . . remove (unlink) files
rmdir . . . . . remove directory
roff . . . . . format text
sh . . . . . shell (command interpreter)
shift . . . . . adjust Shell arguments
size . . . . . size of an object file
sleep . . . . . suspend execution for an interval
sort, usort . . . . . sort or merge files
spell . . . . . find spelling errors
split . . . . . split a file into pieces
strip . . . . . remove symbols and relocation bits
stty . . . . . set typewriter options
tee . . . . . pipe fitting
time . . . . . time a command
tp . . . . . manipulate DECtape and magtape
tr . . . . . transliterate
troff . . . . . format text
tty . . . . . get typewriter name
typo . . . . . find possible typos
uniq . . . . . report repeated lines in a file
wait . . . . . await completion of process
wc . . . . . word count
who . . . . . who is on the system
write . . . . . write to another user
yacc . . . . . yet another compiler-compiler

II. SYSTEM CALLS

intro . . . . . introduction to system calls
break, brk, sbrk . . . . . change core allocation
chdir . . . . . change working directory
chmod . . . . . change mode of file
chown . . . . . change owner and group of a file
close . . . . . close a file
creat . . . . . create a new file
csw . . . . . read console switches
dup . . . . . duplicate an open file descriptor
exec, execl, execv . . . . . execute a file
exit . . . . . terminate process
fork . . . . . spawn new process
fstat . . . . . get status of open file
getgid . . . . . get group identifications
getpid . . . . . get process identification
getuid . . . . . get user identifications
gtty . . . . . get typewriter status
indir . . . . . indirect system call

-

kill . . . . . send signal to a process
link . . . . . link to a file
mknod . . . . . make a directory or a special file
mount . . . . . mount file system
nice . . . . . set program priority
open . . . . . open for reading or writing
pipe . . . . . create an interprocess channel
profil . . . . . execution time profile
ptrace . . . . . process trace
read . . . . . read from file
seek . . . . . move read/write pointer
setgid . . . . . set process group ID
setuid . . . . . set process user ID
signal . . . . . catch or ignore signals
sleep . . . . . stop execution for interval
stat . . . . . get file status
stime . . . . . set time
stty . . . . . set mode of typewriter
sync . . . . . update super-block
time . . . . . get date and time
times . . . . . get process times
umount . . . . . dismount file system
unlink . . . . . remove directory entry
wait . . . . . wait for process to terminate
write . . . . . write on a file

III. SUBROUTINES

abort . . . . . generate an IOT fault
abs, fabs . . . . . absolute value
alloc, free . . . . . core allocator
atan, atan2 . . . . . arc tangent function
atof . . . . . convert ASCII to floating
atoi . . . . . convert ASCII to integer
crypt . . . . . password encoding
ctime, localtime, gmtime . . . . . convert date and time to ASCII
ecvt, fcvt . . . . . output conversion
end, etext, edata . . . . . last locations in program
exp . . . . . exponential function
floor, ceil . . . . . floor and ceiling functions
fmod . . . . . floating modulo function
fptrap . . . . . floating point interpreter
gamma . . . . . log gamma function
getarg, iargc . . . . . get command arguments from Fortran
getc, getw, fopen . . . . . buffered input
getchar . . . . . read character
getpw . . . . . get name from UID
hmul . . . . . high-order product
ierror . . . . . catch Fortran errors
ldiv, lrem . . . . . long division
locv . . . . . long output conversion
log . . . . . natural logarithm
monitor . . . . . prepare execution profile

nargs . . . . . argument count
nlist . . . . . get entries from name list
perror, syserrlist, sysnerr, errno . . . . . system error messages
pow . . . . . floating exponentiation
printf . . . . . formatted print
putc, putw, fcreat, fflush . . . . . buffered output
putchar, flush . . . . . write character
qsort . . . . . quicker sort
rand, srand . . . . . random number generator
reset, setexit . . . . . execute non-local goto
setfil . . . . . specify Fortran file name
sin, cos . . . . . trigonometric functions
sqrt . . . . . square root function
ttyn . . . . . return name of current typewriter

IV. SPECIAL FILES

cat . . . . . phototypesetter interface
dc . . . . . DC-11 communications interface
dh . . . . . DH-11 communications multiplexer
dn . . . . . DN-11 ACU interface
dp . . . . . DP-11 201 data-phone interface
hp . . . . . RH-11/RP04 moving-head disk
hs . . . . . RH11/RS03-RS04 fixed-head disk file
ht . . . . . RH-11/TU-16 magtape interface
kl . . . . . KL-11 or DL-11 asynchronous interface
lp . . . . . line printer
mem, kmem, null . . . . . core memory
pc . . . . . PC-11 paper tape reader/punch
rf . . . . . RF11/RS11 fixed-head disk file
rk . . . . . RK-11/RK03 (or RK05) disk
rp . . . . . RP-11/RP03 moving-head disk
tc . . . . . TC-11/TU56 DECtape
tm . . . . . TM-11/TU-10 magtape interface
tty . . . . . general typewriter interface

V. FILE FORMATS AND CONVENTIONS

a.out . . . . . assembler and link editor output
ar . . . . . archive (library) file format
ascii . . . . . map of ASCII character set
core . . . . . format of core image file
dir . . . . . format of directories
dump . . . . . incremental dump tape format
fs . . . . . format of file system volume
greek . . . . . graphics for extended TTY-37 type-box
group . . . . . group file
mtab . . . . . mounted file system table
passwd . . . . . password file
tabs . . . . . set tab stops
tp . . . . . DEC/mag tape formats
ttys . . . . . typewriter initialization data
utmp . . . . . user information

-

wtmp . . . . . user login history

VI. USER MAINTAINED PROGRAMS

azel . . . . . satellite predictions
bj . . . . . the game of black jack
cal . . . . . print calendar
chess . . . . . the game of chess
col . . . . . filter reverse line feeds
cubic . . . . . three dimensional tic-tac-toe
factor . . . . . discover prime factors of a number
fed . . . . . edit form letter memory
form . . . . . form letter generator
graph . . . . . draw a graph
gsi . . . . . interpret extended character set on GSI terminal
m6 . . . . . general purpose macroprocessor
moo . . . . . guessing game
plot: tek, gsip, vt0 . . . . . graphics filters
primes . . . . . print all primes larger than somewhat
quiz . . . . . test your knowledge
sky . . . . . obtain ephemerides
sno . . . . . Snobol interpreter
speak . . . . . word to voice translator
spline . . . . . interpolate smooth curve
tbl . . . . . format tables for nroff or troff
tmg . . . . . compiler-compiler
ttt . . . . . the game of tic-tac-toe
units . . . . . conversion program
wump . . . . . the game of hunt-the-wumpus

VII. USER MAINTAINED SUBROUTINES

crfork, crexit, crread, crwrite, crexch, crprior . . . . . coroutine scheme
ms . . . . . macros for formatting manuscripts
plot: openpl et al. . . . . graphics interface
salloc . . . . . string allocation and manipulation

VIII. SYSTEM MAINTENANCE

ac . . . . . login accounting
boot procedures . . . . . UNIX startup
chgrp . . . . . change group
chown . . . . . change owner
clri . . . . . clear i-node
crash . . . . . what to do when the system crashes
cron . . . . . clock daemon
dcheck . . . . . file system directory consistency check
df . . . . . disk free
dpd . . . . . data phone daemon
dump . . . . . incremental file system dump
getty . . . . . set typewriter mode
glob . . . . . generate command arguments
icheck . . . . . file system storage consistency check

-

init . . . . . process control initialization
lpd . . . . . line printer daemon
mkfs . . . . . construct a file system
mknod . . . . . build special file
mount . . . . . mount file system
ncheck . . . . . generate names from i-numbers
restor . . . . . incremental file system restore
sa . . . . . Shell accounting
su . . . . . become privileged user
sync . . . . . update the super block
umount . . . . . dismount file system
update . . . . . periodically update the super block
wall . . . . . write to all users

-

PERMUTED INDEX

dp(IV) DP-11 201 data-phone interface
abort(III) generate an IOT fault
abs, fabs(III) absolute value
abs, fabs(III) absolute value
ac(VIII) login accounting
sa(VIII) Shell accounting
dn(IV) DN-11 ACU interface
ac(VIII) login accounting
shift(I) adjust Shell arguments
primes(VI) print all primes larger than somewhat
wall(VIII) write to all users
alloc, free(III) core allocator
salloc(VII) string allocation and manipulation
break, brk, sbrk(II) change core allocation
alloc, free(III) core allocator
plot: openpl et al.(VII) graphics interface
yacc(I) yet another compiler-compiler
write(I) write to another user
a.out(V) assembler and link editor output
bc(I) arbitrary precision interactive language
atan, atan2(III) arc tangent function
ar(I) archive and library maintainer
ar(V) archive (library) file format
nargs(III) argument count
getarg, iargc(III) get command arguments from Fortran
echo(I) echo arguments
glob(VIII) generate command arguments
shift(I) adjust Shell arguments
ar(I) archive and library maintainer
ar(V) archive (library) file format
ascii(V) map of ASCII character set
atof(III) convert ASCII to floating
atoi(III) convert ASCII to integer
gmtime(III) convert date and time to ASCII...ctime, localtime,
ascii(V) map of ASCII character set
as(I) assembler
a.out(V) assembler and link editor output
as(I) assembler
kl(IV) KL-11 or DL-11 asynchronous interface
nice(I) run a command at low priority
atan, atan2(III) arc tangent function
atan, atan2(III) arc tangent function
atof(III) convert ASCII to floating
atoi(III) convert ASCII to integer
wait(I) await completion of process
azel(VI) satellite predictions
bas(I) basic
bas(I) basic
bc(I) arbitrary precision interactive language
su(VIII) become privileged user
strip(I) remove symbols and relocation bits

-

bj(VI) the game of black jack
bj(VI) the game of black jack
sync(VIII) update the super block
update(VIII) periodically update the super block
boot procedures(VIII) UNIX startup
break, brk, sbrk(II) change core allocation
break, brk, sbrk(II) change core allocation
getc, getw, fopen(III) buffered input
putc, putw, fcreat, fflush(III) buffered output
mknod(VIII) build special file
cc(I) C compiler
cdb(I) C debugger
dc(I) desk calculator
cal(VI) print calendar
indir(II) indirect system call
intro(II) introduction to system calls
cal(VI) print calendar
ierror(III) catch Fortran errors
signal(II) catch or ignore signals
cat(I) concatenate and print
cat(IV) phototypesetter interface
cc(I) C compiler
cdb(I) C debugger
floor, ceil(III) floor and ceiling functions
floor, ceil(III) floor and ceiling functions
break, brk, sbrk(II) change core allocation
chgrp(VIII) change group
passwd(I) change login password
chmod(II) change mode of file
chmod(I) change mode
chown(II) change owner and group of a file
chown(VIII) change owner
chdir(I) change working directory
chdir(II) change working directory
pipe(II) create an interprocess channel
gsi(VI) interpret extended character set on GSI terminal
ascii(V) map of ASCII character set
getchar(III) read character
putchar, flush(III) write character
chdir(I) change working directory
chdir(II) change working directory
file system directory consistency check...dcheck(VIII)
file system storage consistency check...icheck(VIII)
chess(VI) the game of chess
chess(VI) the game of chess
chgrp(VIII) change group
chmod(I) change mode
chmod(II) change mode of file
chown(II) change owner and group of a file
chown(VIII) change owner
clri(VIII) clear i-node
cron(VIII) clock daemon
close(II) close a file

-

close(II) close a file
clri(VIII) clear i-node
cmp(I) compare two files
col(VI) filter reverse line feeds
getarg, iargc(III) get command arguments from Fortran
glob(VIII) generate command arguments
nice(I) run a command at low priority
exit(I) terminate command file
nohup(I) run a command immune to hangups
sh(I) shell (command interpreter)
goto(I) command transfer
if(I) conditional command
time(I) time a command
comm(I) print lines common to two files
comm(I) print lines common to two files
dc(IV) DC-11 communications interface
dh(IV) DH-11 communications multiplexer
diff(I) differential file comparator
cmp(I) compare two files
cc(I) C compiler
tmg(VI) compiler-compiler
yacc(I) yet another compiler-compiler
fc(I) Fortran compiler
rc(I) Ratfor compiler
wait(I) await completion of process
cat(I) concatenate and print
if(I) conditional command
dcheck(VIII) file system directory consistency check
icheck(VIII) file system storage consistency check
csw(II) read console switches
mkfs(VIII) construct a file system
ls(I) list contents of directory
init(VIII) process control initialization
units(VI) conversion program
ecvt, fcvt(III) output conversion
locv(III) long output conversion
dd(I) convert and copy a file
atof(III) convert ASCII to floating
atoi(III) convert ASCII to integer
ctime, localtime, gmtime(III) convert date and time to ASCII
dd(I) convert and copy a file
cp(I) copy
break, brk, sbrk(II) change core allocation
alloc, free(III) core allocator
core(V) format of core image file
mem, kmem, null(IV) core memory
core(V) format of core image file
crread, crwrite, crexch, crprior(VII) coroutine scheme...crfork, crexit,
sin, cos(III) trigonometric functions
nargs(III) argument count
wc(I) word count
cp(I) copy
crash(VIII) what to do when the system crashes

-

crash(VIII) what to do when the system crashes
creat(II) create a new file
pipe(II) create an interprocess channel
creat(II) create a new file
cref(I) make cross reference listing
crfork, crexit, crread, crwrite, crexch, crprior(VII) coroutine scheme
scheme...crfork, crexit, crread, crwrite, crexch, crprior(VII) coroutine
coroutine scheme... crfork, crexit, crread, crwrite, crexch, crprior(VII)
cron(VIII) clock daemon
cref(I) make cross reference listing
crfork, crexit, crread, crwrite, crexch, crprior(VII) coroutine scheme
crfork, crexit, crread, crwrite, crexch, crprior(VII) coroutine scheme
crfork, crexit, crread, crwrite, crexch, crprior(VII) coroutine scheme
crypt(III) password encoding
csw(II) read console switches
ASCII... ctime, localtime, gmtime(III) convert date and time to
cubic(VI) three dimensional tic-tac-toe
ttyn(III) return name of current typewriter
spline(VI) interpolate smooth curve
cron(VIII) clock daemon
dpd(VIII) data phone daemon
lpd(VIII) line printer daemon
dpd(VIII) data phone daemon
dp(IV) DP-11 201 data-phone interface
prof(I) display profile data
ttys(V) typewriter initialization data
ctime, localtime, gmtime(III) convert date and time to ASCII
time(II) get date and time
date(I) print and set the date
date(I) print and set the date
db(I) debug
dc(IV) DC-11 communications interface
dcheck(VIII) file system directory consistency check
dc(I) desk calculator
dc(IV) DC-11 communications interface
dd(I) convert and copy a file
db(I) debug
cdb(I) C debugger
tp(V) DEC/mag tape formats
tp(I) manipulate DECtape and magtape
tc(IV) TC-11/TU56 DECtape
dsw(I) delete interactively
mesg(I) permit or deny messages
dup(II) duplicate an open file descriptor
mail(I) send mail to designated users
dc(I) desk calculator
file(I) determine file type
df(VIII) disk free
dh(IV) DH-11 communications multiplexer
dh(IV) DH-11 communications multiplexer
diff(I) differential file comparator
diff(I) differential file comparator
cubic(VI) three dimensional tic-tac-toe

-

dir(V) format of directories
dcheck(VIII) file system directory consistency check
unlink(II) remove directory entry
pwd(I) working directory name
mknod(II) make a directory or a special file
chdir(I) change working directory
chdir(II) change working directory
ls(I) list contents of directory
mkdir(I) make a directory
rmdir(I) remove directory
dir(V) format of directories
factor(VI) discover prime factors of a number
hs(IV) RH11/RS03-RS04 fixed-head disk file
rf(IV) RF11/RS11 fixed-head disk file
df(VIII) disk free
du(I) summarize disk usage
hp(IV) RH-11/RP04 moving-head disk
rk(IV) RK-11/RK03 (or RK05) disk
rp(IV) RP-11/RP03 moving-head disk
umount(II) dismount file system
umount(VIII) dismount file system
prof(I) display profile data
ldiv, lrem(III) long division
kl(IV) KL-11 or DL-11 asynchronous interface
dn(IV) DN-11 ACU interface
dn(IV) DN-11 ACU interface
crash(VIII) what to do when the system crashes
dp(IV) DP-11 201 data-phone interface
dpd(VIII) data phone daemon
dp(IV) DP-11 201 data-phone interface
graph(VI) draw a graph
dsw(I) delete interactively
du(I) summarize disk usage
dump(V) incremental dump tape format
dump(VIII) incremental file system dump
od(I) octal dump
dump(V) incremental dump tape format
dump(VIII) incremental file system dump
dup(II) duplicate an open file descriptor
dup(II) duplicate an open file descriptor
echo(I) echo arguments
echo(I) echo arguments
ecvt, fcvt(III) output conversion
end, etext, edata(III) last locations in program
ed(I) text editor
fed(VI) edit form letter memory
a.out(V) assembler and link editor output
ed(I) text editor
ld(I) link editor
crypt(III) password encoding
end, etext, edata(III) last locations in program
nlist(III) get entries from name list
unlink(II) remove directory entry

-

sky(VI) obtain ephemerides
eqn(I) typeset mathematics
perror, syserrlist, sysnerr, errno(III) system error messages
sysnerr, errno(III) system error messages...perror, syserrlist,
ierror(III) catch Fortran errors
spell(I) find spelling errors
plot: openpl et al.(VII) graphics interface
end, etext, edata(III) last locations in program
pfe(I) print floating exception
exec, execl, execv(II) execute a file
exec, execl, execv(II) execute a file
exec, execl, execv(II) execute a file
reset, setexit(III) execute non-local goto
sleep(I) suspend execution for an interval
sleep(II) stop execution for interval
monitor(III) prepare execution profile
profil(II) execution time profile
exec, execl, execv(II) execute a file
exit(I) terminate command file
exit(II) terminate process
exp(III) exponential function
exp(III) exponential function
pow(III) floating exponentiation
gsi(VI) interpret extended character set on GSI terminal
greek(V) graphics for extended TTY-37 type-box
abs, fabs(III) absolute value
factor(VI) discover prime factors of a number
factor(VI) discover prime factors of a number
abort(III) generate an IOT fault
fc(I) Fortran compiler
putc, putw, fcreat, fflush(III) buffered output
ecvt, fcvt(III) output conversion
fed(VI) edit form letter memory
col(VI) filter reverse line feeds
putc, putw, fcreat, fflush(III) buffered output
diff(I) differential file comparator
dup(II) duplicate an open file descriptor
grep(I) search a file for a pattern
ar(V) archive (library) file format
split(I) split a file into pieces
setfil(III) specify Fortran file name
stat(II) get file status
dcheck(VIII) file system directory consistency check
dump(VIII) incremental file system dump
restor(VIII) incremental file system restore
icheck(VIII) file system storage consistency check
mtab(V) mounted file system table
fs(V) format of file system volume
mkfs(VIII) construct a file system
mount(II) mount file system
mount(VIII) mount file system
umount(II) dismount file system
umount(VIII) dismount file system

-

file(I) determine file type
chmod(II) change mode of file
chown(II) change owner and group of a file
close(II) close a file
core(V) format of core image file
creat(II) create a new file
dd(I) convert and copy a file
exec, execl, execv(II) execute a file
exit(I) terminate command file
fstat(II) get status of open file
group(V) group file
hs(IV) RH11/RS03-RS04 fixed-head disk file
file(I) determine file type
link(II) link to a file
mknod(II) make a directory or a special file
mknod(VIII) build special file
mv(I) move or rename a file
passwd(V) password file
pr(I) print file
read(II) read from file
rev(I) reverse lines of a file
rf(IV) RF11/RS11 fixed-head disk file
cmp(I) compare two files
comm(I) print lines common to two files
find(I) find files
size(I) size of an object file
rm(I) remove (unlink) files
sort, usort(I) sort or merge files
uniq(I) report repeated lines in a file
write(II) write on a file
col(VI) filter reverse line feeds
plot: tek, gsip, vt0(VI) graphics filters
find(I) find files
typo(I) find possible typos
spell(I) find spelling errors
find(I) find files
tee(I) pipe fitting
hs(IV) RH11/RS03-RS04 fixed-head disk file
rf(IV) RF11/RS11 fixed-head disk file
pfe(I) print floating exception
pow(III) floating exponentiation
fmod(III) floating modulo function
fptrap(III) floating point interpreter
atof(III) convert ASCII to floating
floor, ceil(III) floor and ceiling functions
floor, ceil(III) floor and ceiling functions
putchar, flush(III) write character
fmod(III) floating modulo function
getc, getw, fopen(III) buffered input
fork(II) spawn new process
form(VI) form letter generator
fed(VI) edit form letter memory
core(V) format of core image file

-

dir(V) format of directories
fs(V) format of file system volume
tbl(VI) format tables for nroff or troff
nroff(I) format text
roff(I) format text
troff(I) format text
ar(V) archive (library) file format
dump(V) incremental dump tape format
tp(V) DEC/mag tape formats
printf(III) formatted print
ms(VII) macros for formatting manuscripts
form(VI) form letter generator
fc(I) Fortran compiler
ierror(III) catch Fortran errors
setfil(III) specify Fortran file name
iargc(III) get command arguments from Fortran...getarg,
fptrap(III) floating point interpreter
df(VIII) disk free
alloc, free(III) core allocator
read(II) read from file
getarg, iargc(III) get command arguments from Fortran
ncheck(VIII) generate names from i-numbers
nlist(III) get entries from name list
getpw(III) get name from UID
fstat(II) get status of open file
fs(V) format of file system volume
atan, atan2(III) arc tangent function
exp(III) exponential function
fmod(III) floating modulo function
gamma(III) log gamma function
floor, ceil(III) floor and ceiling functions
sqrt(III) square root function
sin, cos(III) trigonometric functions
bj(VI) the game of black jack
chess(VI) the game of chess
wump(VI) the game of hunt-the-wumpus
ttt(VI) the game of tic-tac-toe
moo(VI) guessing game
gamma(III) log gamma function
gamma(III) log gamma function
m6(VI) general purpose macroprocessor
tty(IV) general typewriter interface
abort(III) generate an IOT fault
glob(VIII) generate command arguments
ncheck(VIII) generate names from i-numbers
form(VI) form letter generator
rand, srand(III) random number generator
getarg, iargc(III) get command arguments from Fortran
time(II) get date and time
nlist(III) get entries from name list
stat(II) get file status
getgid(II) get group identifications
getpw(III) get name from UID

-

getpid(II) get process identification
times(II) get process times
fstat(II) get status of open file
tty(I) get typewriter name
gtty(II) get typewriter status
getuid(II) get user identifications
getarg, iargc(III) get command arguments from Fortran
getc, getw, fopen(III) buffered input
getchar(III) read character
getgid(II) get group identifications
getpid(II) get process identification
getpw(III) get name from UID
getty(VIII) set typewriter mode
getuid(II) get user identifications
getc, getw, fopen(III) buffered input
glob(VIII) generate command arguments
ctime, localtime, gmtime(III) convert date and time to ASCII
goto(I) command transfer
reset, setexit(III) execute non-local goto
graph(VI) draw a graph
plot: tek, gsip, vt0(VI) graphics filters
greek(V) graphics for extended TTY-37 type-box
plot: openpl et al.(VII) graphics interface
graph(VI) draw a graph
greek(V) graphics for extended TTY-37 type-box
grep(I) search a file for a pattern
group(V) group file
getgid(II) get group identifications
setgid(II) set process group ID
chown(II) change owner and group of a file
chgrp(VIII) change group
newgrp(I) log in to a new group
group(V) group file
gsi(VI) interpret extended character set on GSI terminal
plot: tek, gsip, vt0(VI) graphics filters
gsi(VI) interpret extended character set on GSI terminal
gtty(II) get typewriter status
moo(VI) guessing game
nohup(I) run a command immune to hangups
hmul(III) high-order product
wtmp(V) user login history
hmul(III) high-order product
hp(IV) RH-11/RP04 moving-head disk
hs(IV) RH11/RS03-RS04 fixed-head disk file
ht(IV) RH-11/TU-16 magtape interface
wump(VI) the game of hunt-the-wumpus
getarg, iargc(III) get command arguments from Fortran
icheck(VIII) file system storage consistency check
getpid(II) get process identification
getgid(II) get group identifications
getuid(II) get user identifications
setgid(II) set process group ID
setuid(II) set process user ID

-

ierror(III) catch Fortran errors
if(I) conditional command
signal(II) catch or ignore signals
core(V) format of core image file
nohup(I) run a command immune to hangups
uniq(I) report repeated lines in a file
end, etext, edata(III) last locations in program
newgrp(I) log in to a new group
dump(V) incremental dump tape format
dump(VIII) incremental file system dump
restor(VIII) incremental file system restore
indir(II) indirect system call
indir(II) indirect system call
utmp(V) user information
ttys(V) typewriter initialization data
init(VIII) process control initialization
init(VIII) process control initialization
clri(VIII) clear i-node
getc, getw, fopen(III) buffered input
atoi(III) convert ASCII to integer
bc(I) arbitrary precision interactive language
dsw(I) delete interactively
cat(IV) phototypesetter interface
dc(IV) DC-11 communications interface
dn(IV) DN-11 ACU interface
dp(IV) DP-11 201 data-phone interface
ht(IV) RH-11/TU-16 magtape interface
kl(IV) KL-11 or DL-11 asynchronous interface
plot: openpl et al.(VII) graphics interface
tm(IV) TM-11/TU-10 magtape interface
tty(IV) general typewriter interface
spline(VI) interpolate smooth curve
gsi(VI) interpret extended character set on GSI terminal
fptrap(III) floating point interpreter
sh(I) shell (command interpreter)
sno(VI) Snobol interpreter
pipe(II) create an interprocess channel
sleep(I) suspend execution for an interval
sleep(II) stop execution for interval
split(I) split a file into pieces
intro(II) introduction to system calls
intro(II) introduction to system calls
ncheck(VIII) generate names from i-numbers
abort(III) generate an IOT fault
bj(VI) the game of black jack
kill(I) terminate a process
kill(II) send signal to a process
kl(IV) KL-11 or DL-11 asynchronous interface
kl(IV) KL-11 or DL-11 asynchronous interface
mem, kmem, null(IV) core memory
quiz(VI) test your knowledge
bc(I) arbitrary precision interactive language
primes(VI) print all primes larger than somewhat

-

end, etext, edata(III) last locations in program
ld(I) link editor
ldiv, lrem(III) long division
form(VI) form letter generator
fed(VI) edit form letter memory
ar(V) archive (library) file format
ar(I) archive and library maintainer
col(VI) filter reverse line feeds
lpd(VIII) line printer daemon
lp(IV) line printer
opr(I) off line print
comm(I) print lines common to two files
uniq(I) report repeated lines in a file
rev(I) reverse lines of a file
a.out(V) assembler and link editor output
ld(I) link editor
link(II) link to a file
link(II) link to a file
ln(I) make a link
ls(I) list contents of directory
cref(I) make cross reference listing
nlist(III) get entries from name list
nm(I) print name list
ln(I) make a link
ctime, localtime, gmtime(III) convert date and time to ASCII
end, etext, edata(III) last locations in program
locv(III) long output conversion
gamma(III) log gamma function
newgrp(I) log in to a new group
log(III) natural logarithm
log(III) natural logarithm
ac(VIII) login accounting
wtmp(V) user login history
passwd(I) change login password
login(I) sign onto UNIX
ldiv, lrem(III) long division
locv(III) long output conversion
nice(I) run a command at low priority
lpd(VIII) line printer daemon
lp(IV) line printer
ldiv, lrem(III) long division
ls(I) list contents of directory
m6(VI) general purpose macroprocessor
m6(VI) general purpose macroprocessor
ms(VII) macros for formatting manuscripts
ht(IV) RH-11/TU-16 magtape interface
tm(IV) TM-11/TU-10 magtape interface
tp(I) manipulate DECtape and magtape
mail(I) send mail to designated users
mail(I) send mail to designated users
ar(I) archive and library maintainer
mknod(II) make a directory or a special file
mkdir(I) make a directory

-

ln(I) make a link
cref(I) make cross reference listing
man(I) run off section of UNIX manual
tp(I) manipulate DECtape and magtape
salloc(VII) string allocation and manipulation
man(I) run off section of UNIX manual
ms(VII) macros for formatting manuscripts
ascii(V) map of ASCII character set
neqn(I) typeset mathematics on terminal
eqn(I) typeset mathematics
mem, kmem, null(IV) core memory
fed(VI) edit form letter memory
mem, kmem, null(IV) core memory
sort, usort(I) sort or merge files
mesg(I) permit or deny messages
mesg(I) permit or deny messages
sysnerr, errno(III) system error messages...perror, syserrlist,
mkdir(I) make a directory
mkfs(VIII) construct a file system
mknod(II) make a directory or a special file
mknod(VIII) build special file
chmod(II) change mode of file
stty(II) set mode of typewriter
chmod(I) change mode
getty(VIII) set typewriter mode
fmod(III) floating modulo function
monitor(III) prepare execution profile
moo(VI) guessing game
mount(II) mount file system
mount(VIII) mount file system
mtab(V) mounted file system table
mount(II) mount file system
mount(VIII) mount file system
mv(I) move or rename a file
seek(II) move read/write pointer
hp(IV) RH-11/RP04 moving-head disk
rp(IV) RP-11/RP03 moving-head disk
ms(VII) macros for formatting manuscripts
mtab(V) mounted file system table
dh(IV) DH-11 communications multiplexer
mv(I) move or rename a file
getpw(III) get name from UID
nlist(III) get entries from name list
nm(I) print name list
ttyn(III) return name of current typewriter
pwd(I) working directory name
ncheck(VIII) generate names from i-numbers
setfil(III) specify Fortran file name
tty(I) get typewriter name
nargs(III) argument count
log(III) natural logarithm
ncheck(VIII) generate names from i-numbers
neqn(I) typeset mathematics on terminal

-

creat(II) create a new file
newgrp(I) log in to a new group
fork(II) spawn new process
newgrp(I) log in to a new group
nice(I) run a command at low priority
nice(II) set program priority
nlist(III) get entries from name list
nm(I) print name list
nohup(I) run a command immune to hangups
reset, setexit(III) execute non-local goto
tbl(VI) format tables for nroff or troff
nroff(I) format text
mem, kmem, null(IV) core memory
rand, srand(III) random number generator
factor(VI) discover prime factors of a number
size(I) size of an object file
sky(VI) obtain ephemerides
od(I) octal dump
od(I) octal dump
opr(I) off line print
man(I) run off section of UNIX manual
login(I) sign onto UNIX
dup(II) duplicate an open file descriptor
fstat(II) get status of open file
open(II) open for reading or writing
open(II) open for reading or writing
plot: openpl et al.(VII) graphics interface
opr(I) off line print
stty(I) set typewriter options
rk(IV) RK-11/RK03 (or RK05) disk
ecvt, fcvt(III) output conversion
locv(III) long output conversion
a.out(V) assembler and link editor output
putc, putw, fcreat, fflush(III) buffered output
chown(II) change owner and group of a file
chown(VIII) change owner
pc(IV) PC-11 paper tape reader/punch
passwd(I) change login password
passwd(V) password file
crypt(III) password encoding
passwd(V) password file
passwd(I) change login password
grep(I) search a file for a pattern
pc(IV) PC-11 paper tape reader/punch
pc(IV) PC-11 paper tape reader/punch
update(VIII) periodically update the super block
mesg(I) permit or deny messages
error messages... perror, syserrlist, sysnerr, errno(III) system
pfe(I) print floating exception
dpd(VIII) data phone daemon
cat(IV) phototypesetter interface
split(I) split a file into pieces
tee(I) pipe fitting

-

pipe(II) create an interprocess channel
plot: openpl et al.(VII) graphics interface
plot: tek, gsip, vt0(VI) graphics filters
fptrap(III) floating point interpreter
seek(II) move read/write pointer
typo(I) find possible typos
pow(III) floating exponentiation
bc(I) arbitrary precision interactive language
azel(VI) satellite predictions
monitor(III) prepare execution profile
pr(I) print file
factor(VI) discover prime factors of a number
primes(VI) print all primes larger than somewhat
primes(VI) print all primes larger than somewhat
primes(VI) print all primes larger than somewhat
date(I) print and set the date
cal(VI) print calendar
pr(I) print file
pfe(I) print floating exception
comm(I) print lines common to two files
nm(I) print name list
cat(I) concatenate and print
lpd(VIII) line printer daemon
lp(IV) line printer
printf(III) formatted print
opr(I) off line print
printf(III) formatted print
nice(I) run a command at low priority
nice(II) set program priority
su(VIII) become privileged user
boot procedures(VIII) UNIX startup
init(VIII) process control initialization
setgid(II) set process group ID
getpid(II) get process identification
ps(I) process status
times(II) get process times
wait(II) wait for process to terminate
ptrace(II) process trace
setuid(II) set process user ID
exit(II) terminate process
fork(II) spawn new process
kill(I) terminate a process
kill(II) send signal to a process
wait(I) await completion of process
hmul(III) high-order product
prof(I) display profile data
prof(I) display profile data
monitor(III) prepare execution profile
profil(II) execution time profile
profil(II) execution time profile
nice(II) set program priority
end, etext, edata(III) last locations in program
units(VI) conversion program

-

ps(I) process status
ptrace(II) process trace
m6(VI) general purpose macroprocessor
putc, putw, fcreat, fflush(III) buffered output
putchar, flush(III) write character
putc, putw, fcreat, fflush(III) buffered output
pwd(I) working directory name
qsort(III) quicker sort
qsort(III) quicker sort
quiz(VI) test your knowledge
rand, srand(III) random number generator
rand, srand(III) random number generator
rc(I) Ratfor compiler
rc(I) Ratfor compiler
getchar(III) read character
csw(II) read console switches
read(II) read from file
pc(IV) PC-11 paper tape reader/punch
read(II) read from file
open(II) open for reading or writing
seek(II) move read/write pointer
cref(I) make cross reference listing
strip(I) remove symbols and relocation bits
unlink(II) remove directory entry
rmdir(I) remove directory
strip(I) remove symbols and relocation bits
rm(I) remove (unlink) files
mv(I) move or rename a file
uniq(I) report repeated lines in a file
uniq(I) report repeated lines in a file
reset, setexit(III) execute non-local goto
restor(VIII) incremental file system restore
restor(VIII) incremental file system restore
ttyn(III) return name of current typewriter
col(VI) filter reverse line feeds
rev(I) reverse lines of a file
rev(I) reverse lines of a file
rf(IV) RF11/RS11 fixed-head disk file
rf(IV) RF11/RS11 fixed-head disk file
hp(IV) RH-11/RP04 moving-head disk
hs(IV) RH11/RS03-RS04 fixed-head disk file
ht(IV) RH-11/TU-16 magtape interface
rk(IV) RK-11/RK03 (or RK05) disk
rk(IV) RK-11/RK03 (or RK05) disk
rk(IV) RK-11/RK03 (or RK05) disk
rmdir(I) remove directory
rm(I) remove (unlink) files
roff(I) format text
sqrt(III) square root function
rp(IV) RP-11/RP03 moving-head disk
rp(IV) RP-11/RP03 moving-head disk
nice(I) run a command at low priority
nohup(I) run a command immune to hangups

-

man(I) run off section of UNIX manual
salloc(VII) string allocation and manipulation
azel(VI) satellite predictions
sa(VIII) Shell accounting
break, brk, sbrk(II) change core allocation
crwrite, crexch, crprior(VII) coroutine scheme...crfork, crexit, crread,
grep(I) search a file for a pattern
man(I) run off section of UNIX manual
seek(II) move read/write pointer
mail(I) send mail to designated users
kill(II) send signal to a process
stty(II) set mode of typewriter
gsi(VI) interpret extended character set on GSI terminal
setgid(II) set process group ID
setuid(II) set process user ID
nice(II) set program priority
tabs(V) set tab stops
date(I) print and set the date
stime(II) set time
getty(VIII) set typewriter mode
stty(I) set typewriter options
ascii(V) map of ASCII character set
reset, setexit(III) execute non-local goto
setfil(III) specify Fortran file name
setgid(II) set process group ID
setuid(II) set process user ID
sa(VIII) Shell accounting
shift(I) adjust Shell arguments
sh(I) shell (command interpreter)
sh(I) shell (command interpreter)
shift(I) adjust Shell arguments
login(I) sign onto UNIX
kill(II) send signal to a process
signal(II) catch or ignore signals
signal(II) catch or ignore signals
sin, cos(III) trigonometric functions
size(I) size of an object file
size(I) size of an object file
sky(VI) obtain ephemerides
sleep(I) suspend execution for an interval
sleep(II) stop execution for interval
spline(VI) interpolate smooth curve
sno(VI) Snobol interpreter
sno(VI) Snobol interpreter
primes(VI) print all primes larger than somewhat
sort, usort(I) sort or merge files
sort, usort(I) sort or merge files
qsort(III) quicker sort
fork(II) spawn new process
speak(VI) word to voice translator
mknod(II) make a directory or a special file
mknod(VIII) build special file
setfil(III) specify Fortran file name

-

spell(I) find spelling errors
spell(I) find spelling errors
spline(VI) interpolate smooth curve
split(I) split a file into pieces
split(I) split a file into pieces
sqrt(III) square root function
sqrt(III) square root function
rand, srand(III) random number generator
boot procedures(VIII) UNIX startup
stat(II) get file status
fstat(II) get status of open file
gtty(II) get typewriter status
ps(I) process status
stat(II) get file status
stime(II) set time
sleep(II) stop execution for interval
tabs(V) set tab stops
icheck(VIII) file system storage consistency check
salloc(VII) string allocation and manipulation
strip(I) remove symbols and relocation bits
stty(I) set typewriter options
stty(II) set mode of typewriter
du(I) summarize disk usage
sync(VIII) update the super block
update(VIII) periodically update the super block
sync(II) update super-block
sleep(I) suspend execution for an interval
su(VIII) become privileged user
csw(II) read console switches
strip(I) remove symbols and relocation bits
sync(II) update super-block
sync(VIII) update the super block
messages...perror, syserrlist, sysnerr, errno(III) system error
perror, syserrlist, sysnerr, errno(III) system error messages
indir(II) indirect system call
intro(II) introduction to system calls
crash(VIII) what to do when the system crashes
dcheck(VIII) file system directory consistency check
dump(VIII) incremental file system dump
syserrlist, sysnerr, errno(III) system error messages...perror,
restor(VIII) incremental file system restore
icheck(VIII) file system storage consistency check
mtab(V) mounted file system table
fs(V) format of file system volume
mkfs(VIII) construct a file system
mount(II) mount file system
mount(VIII) mount file system
umount(II) dismount file system
umount(VIII) dismount file system
who(I) who is on the system
tabs(V) set tab stops
mtab(V) mounted file system table
tbl(VI) format tables for nroff or troff

-

tabs(V) set tab stops
atan, atan2(III) arc tangent function
dump(V) incremental dump tape format
tp(V) DEC/mag tape formats
pc(IV) PC-11 paper tape reader/punch
tbl(VI) format tables for nroff or troff
tc(IV) TC-11/TU56 DECtape
tc(IV) TC-11/TU56 DECtape
tee(I) pipe fitting
plot: tek, gsip, vt0(VI) graphics filters
interpret extended character set on GSI terminal...gsi(VI)
neqn(I) typeset mathematics on terminal
kill(I) terminate a process
exit(I) terminate command file
exit(II) terminate process
wait(II) wait for process to terminate
quiz(VI) test your knowledge
ed(I) text editor
nroff(I) format text
roff(I) format text
troff(I) format text
primes(VI) print all primes larger than somewhat
cubic(VI) three dimensional tic-tac-toe
cubic(VI) three dimensional tic-tac-toe
ttt(VI) the game of tic-tac-toe
time(I) time a command
profil(II) execution time profile
localtime, gmtime(III) convert date and time to ASCII...ctime,
time(I) time a command
time(II) get date and time
times(II) get process times
stime(II) set time
times(II) get process times
time(II) get date and time
tm(IV) TM-11/TU-10 magtape interface
tmg(VI) compiler-compiler
tm(IV) TM-11/TU-10 magtape interface
tp(I) manipulate DECtape and magtape
tp(V) DEC/mag tape formats
ptrace(II) process trace
goto(I) command transfer
speak(VI) word to voice translator
tr(I) transliterate
tr(I) transliterate
sin, cos(III) trigonometric functions
troff(I) format text
tbl(VI) format tables for nroff or troff
ttt(VI) the game of tic-tac-toe
greek(V) graphics for extended TTY-37 type-box
tty(I) get typewriter name
tty(IV) general typewriter interface
ttyn(III) return name of current typewriter
ttys(V) typewriter initialization data

-

cmp(I) compare two files
comm(I) print lines common to two files
greek(V) graphics for extended TTY-37 type-box
file(I) determine file type
neqn(I) typeset mathematics on terminal
eqn(I) typeset mathematics
ttys(V) typewriter initialization data
tty(IV) general typewriter interface
getty(VIII) set typewriter mode
tty(I) get typewriter name
stty(I) set typewriter options
gtty(II) get typewriter status
stty(II) set mode of typewriter
ttyn(III) return name of current typewriter
typo(I) find possible typos
typo(I) find possible typos
getpw(III) get name from UID
umount(II) dismount file system
umount(VIII) dismount file system
uniq(I) report repeated lines in a file
units(VI) conversion program
man(I) run off section of UNIX manual
boot procedures(VIII) UNIX startup
login(I) sign onto UNIX
rm(I) remove (unlink) files
unlink(II) remove directory entry
sync(II) update super-block
sync(VIII) update the super block
update(VIII) periodically update the super block
update(VIII) periodically update the super block
du(I) summarize disk usage
getuid(II) get user identifications
setuid(II) set process user ID
utmp(V) user information
wtmp(V) user login history
mail(I) send mail to designated users
su(VIII) become privileged user
wall(VIII) write to all users
write(I) write to another user
sort, usort(I) sort or merge files
utmp(V) user information
abs, fabs(III) absolute value
speak(VI) word to voice translator
fs(V) format of file system volume
plot: tek, gsip, vt0(VI) graphics filters
wait(II) wait for process to terminate
wait(I) await completion of process
wait(II) wait for process to terminate
wall(VIII) write to all users
wc(I) word count
crash(VIII) what to do when the system crashes
crash(VIII) what to do when the system crashes
who(I) who is on the system

-

	who(I) who is on the system
wc(I)	word count
speak(VI)	word to voice translator
pwd(I)	working directory name
chdir(I) change	working directory
chdir(II) change	working directory
putchar, flush(III)	write character
write(II)	write on a file
wall(VIII)	write to all users
write(I)	write to another user
	write(I) write to another user
	write(II) write on a file
open(II) open for reading or	writing
	wtmp(V) user login history
	wump(VI) the game of hunt-the-wumpus
	yacc(I) yet another compiler-compiler
yacc(I)	yet another compiler-compiler
quiz(VI) test	your knowledge