

# UNIX PROGRAMMER'S MANUAL

*Fourth Edition*

*K. Thompson*

*D. M. Ritchie*

*November, 1973*

Copyright © 1972, 1973  
Bell Telephone Laboratories, Inc.

No part of this document may be reproduced,  
or distributed outside the Laboratories, without  
the written permission of Bell Telephone Laboratories.

Copyright © 1972, 1973  
Bell Telephone Laboratories, Incorporated

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 Fourth Edition

In the months since the last appearance of this manual, many changes have occurred both in the system itself and in the way it is used. The most important changes result from a complete rewrite of the UNIX system in the C language. There have also been substantial changes in much of the system software. It is these changes, of course, which mandated the new edition of this manual.

The number of UNIX installations is now above 20, and many more are expected. None of these has exactly the same complement of hardware or software. Therefore, at any particular installation, it is quite possible that this manual will give inappropriate information. In particular, *the information in this manual applies only to UNIX systems which operate under the C language versions of the system.* Installations which use older versions of UNIX will find earlier editions of this manual more appropriate to their situation.

Even in installations which have the latest versions of the operating system, not all the software and other facilities mentioned herein will be available. For example, the typesetter, voice response unit, and voice synthesizer are hardly universally available devices; also, some of the UNIX software has not been released for use outside the Bell System.

The authors are grateful to L. L. Cherry, M. E. Lesk, E. N. Pinson, and C. S. Roberts 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” 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
- VI. User-maintained programs
- VII. Miscellaneous
- 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). This directory is searched automatically by the command line interpreter. Some programs also reside in */usr/bin*, to save space in */bin*. Some programs classified as commands are located elsewhere; this fact is indicated in the appropriate sections.

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 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 (section VI) 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 author should be consulted for information.

The miscellaneous section (VII) gathers odds and ends.

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.

*Logging in.* You must call UNIX from an appropriate terminal. UNIX supports ASCII terminals typified by the TTY 37, the GE Terminet 300, the Memorex 1240, 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.

*TTY 37 terminal:* UNIX will type out “login: ”; 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).

*300-baud terminals:* Such terminals include the GE Terminet 300, most display terminals, Execuport, TI, 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. (Note that this switch will often have to be changed since many other systems require half-duplex). When a connection is established, a few garbage characters are typed (the login message at the wrong speed). Depress the “break” key; this is a speed-independent signal to UNIX that a 300-baud terminal is in use. UNIX will type “login: ” at the correct speed; you type your user name, followed by the “return” key. Henceforth, the “return”, “new line”, or “linefeed” keys will give exactly the same results.

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* (VII), which discusses the login sequence in more detail, and *dc* (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. (We reassure you that this doesn’t happen often.)

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* (VII)).

Section *dc* (IV) discusses typewriter I/O more fully. Section *kl* (IV) discusses the console typewriter.

*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/flex* refers to the file *flex* 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, to appear in the Communications of the ACM; a version is also available from the same source as this manual. 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 the assembly-language 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.

The next command you will need is *db* (I). As a debugger, *db* is better than average for assembly-language programs, marginally useful for C programs (when completed, *cdb* (I) will be a boon), and virtually useless 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
cat	concatenate and print
catsim	phototypesetter simulator
cc	C compiler
cdb	C debugger
chdir	change working directory
chmod	change mode
chown	change owner
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
dsw	delete interactively
du	summarize disk usage
echo	echo arguments
ed	editor
exit	terminate command file
fc	fortran compiler
fed	edit associative memory for form letter
file	determine format of file
form	form letter generator
goto	command transfer
grep	search a file for a pattern
if	conditional command
kill	do in an unwanted process
ld	link editor
ln	make a link
login	sign onto UNIX
ls	list contents of directory
mail	send mail to another user
man	run off section of UNIX manual
merge	merge several files
mesg	permit or deny messages
mkdir	make a directory
mv	move or rename a file
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	set login password
pfe	print floating exception
plot	make a graph

pr	print file
proof	compare two text files
ps	process status
rew	rewind tape
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
sno	Snobol interpreter
sort	sort a file
speak	word to voice translator
split	split a file into pieces
strip	remove symbols and relocation bits
stty	set teletype options
sum	sum file
time	time a command
tp	manipulate DEctape and magtape
tr	transliterate
troff	format text
tss	interface to MH-TSS
tty	get typewriter name
type	type on 2741
typo	find possible typos
uniq	report repeated lines in a file
wait	await completion of process
wc	get (English) word count
who	who is on the system
write	write to another user

## II. SYSTEM CALLS

break	set program break
chdir	change working directory
chmod	change mode of file
chown	change owner
close	close a file
creat	create a new file
csw	read console switches
dup	duplicate an open file descriptor
exec	execute a file
exit	terminate process
fork	spawn new process
fstat	get status of open file
getgid	get group identification
getuid	get user identification
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 a pipe
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 die
write	. . . . .	write on a file

### III. SUBROUTINES

atan	. . . . .	arc tangent function
atof	. . . . .	ascii to floating
compar	. . . . .	default comparison routine for qsort
crypt	. . . . .	password encoding
ctime	. . . . .	convert date and time to ASCII
ecvt	. . . . .	output conversion
exp	. . . . .	exponential function
fptrap	. . . . .	floating point interpreter
gerts	. . . . .	Gerts communication over 201
getarg	. . . . .	get command arguments from Fortran
getc	. . . . .	buffered input
getchar	. . . . .	read character
getpw	. . . . .	get name from UID
hmul	. . . . .	high-order product
hypot	. . . . .	calculate hypotenuse
ierror	. . . . .	catch Fortran errors
ldiv	. . . . .	long division
log	. . . . .	natural logarithm
mesg	. . . . .	write message on typewriter
nargs	. . . . .	argument count
nlist	. . . . .	get entries from name list
perror	. . . . .	system error messages
pow	. . . . .	floating exponentiation
printf	. . . . .	formatted print
putc	. . . . .	buffered output
putchar	. . . . .	write character
qsort	. . . . .	quicker sort
rand	. . . . .	random number generator
reset	. . . . .	execute non-local goto
setfil	. . . . .	specify Fortran file name

sin	.....	sine, cosine
sqrt	.....	square root function
switch	.....	switch on value
ttyn	.....	return name of current typewriter
vt	.....	display (vt01) interface

#### IV. SPECIAL FILES

cat	.....	phototypesetter interface
da	.....	voice response unit
dc	.....	DC-11 communications interface
dn	.....	dn11 ACU interface
dp	.....	dp11 201 data-phone interface
kl	.....	KL-11/TTY-33 console typewriter
mem	.....	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
tiu	.....	Spider interface
tm	.....	TM-11/TU-10 magtape interface
vs	.....	voice synthesizer interface
vt	.....	11/20 (vt01) interface

#### V. FILE FORMATS

a.out	.....	assembler and link editor output
ar	.....	archive (library) file format
core	.....	format of core image file
dir	.....	format of directories
fs	.....	format of file system volume
passwd	.....	password file
tp	.....	DEC/mag tape formats
utmp	.....	user information
wtmp	.....	user login history

#### VI. USER MAINTAINED PROGRAMS

azel	.....	obtain satellite predictions
bj	.....	the game of black jack
cal	.....	print calendar
chess	.....	the game of chess
cubic	.....	three dimensional tic-tac-toe
factor	.....	discover prime factors of a number
hyphen	.....	find hyphenated words
m6	.....	general purpose macro processor
maze	.....	generate a maze problem
moo	.....	guessing game
ov	.....	overlay pages
ptx	.....	permuted index
sfs	.....	structured file scanner
sky	.....	obtain ephemerides

spline . . . . . interpolate smooth curve  
 tmg . . . . . compiler-compiler  
 ttt . . . . . tic-tac-toe  
 wump . . . . . hunt the wumpus  
 yacc . . . . . yet another compiler-compiler

## VII. MISCELLANEOUS

ascii . . . . . map of ASCII character set  
 dpd . . . . . spawn data phone daemon  
 getty . . . . . set typewriter mode  
 glob . . . . . generate command arguments  
 greek . . . . . graphics for extended ascii type-box  
 init . . . . . process control initialization  
 msh . . . . . mini-shell  
 tabs . . . . . set tab stops  
 tmheader . . . . . TM cover sheet  
 vs . . . . . voice synthesizer code

## VIII. SYSTEM MAINTAINANCE

20boot . . . . . install new 11/20 system  
 boot procedures . . . . . UNIX startup  
 check . . . . . file system consistency check  
 clri . . . . . clear i-node  
 df . . . . . disk free  
 dump . . . . . incremental file system dump  
 ino . . . . . get the i-number of a file  
 mkfs . . . . . construct a file system  
 mknod . . . . . build special file  
 mount . . . . . mount file system  
 reloc . . . . . relocate object files  
 restor . . . . . incremental file system restore  
 su . . . . . become privileged user  
 sync . . . . . update the super block  
 umount . . . . . dismount file system  
 update . . . . . periodically update the super block

## PERMUTED INDEX

20boot(VIII) install new 11/20 system  
     vt(IV) 11/20 (vt01) interface  
     dp(IV) dp11 201 data-phone interface  
 gerts(III) Gerts communication over 201  
     20boot(VIII) install new 11/20 system  
     type(I) type on 2741  
     dn(IV) dn11 ACU interface  
     shift(I) adjust Shell arguments  
     yacc(VI) yet another compiler-compiler  
 mail(I) send mail to another user  
 write(I) write to another user  
     a.out(V) assembler and link editor output  
     atan(III) arc tangent function  
     ar(I) archive and library maintainer  
     ar(V) archive (library) file format  
     nargs(III) argument count  
 getarg(III) get command arguments from Fortran  
     echo(I) echo arguments  
 glob(VII) generate command arguments  
     shift(I) adjust Shell arguments  
     ar(I) archive and library maintainer  
     ar(V) archive (library) file format  
     ascii(VII) map of ASCII character set  
     atof(III) ascii to floating  
 greek(VII) graphics for extended ascii type-box  
 ctime(III) convert date and time to ASCII  
     ascii(VII) map of ASCII character set  
     as(I) assembler  
     a.out(V) assembler and link editor output  
     as(I) assembler  
     fed(I) edit associative memory for form letter  
 nice(I) run a command at low priority  
     atan(III) arc tangent function  
     atof(III) ascii to floating  
     wait(I) await completion of process  
     azel(VI) obtain satellite predictions  
     bas(I) basic  
     bas(I) basic  
     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(II) set program break  
     break(II) set program break  
     getc(III) buffered input  
     putc(III) buffered output  
     mknod(VIII) build special file  
     cc(I) C compiler

	cdb(I)	C debugger
	hypot(III)	calculate hypotenuse
	dc(I) desk	calculator
	cal(VI) print	calendar
indir(II)	indirect system	call
		cal(VI) print calendar
	ierror(III)	catch Fortran errors
	signal(II)	catch or ignore signals
		cat(I) concatenate and print
		cat(IV) phototypesetter interface
		catsim(I) phototypesetter simulator
		cc(I) C compiler
		cdb(I) C debugger
	chmod(II)	change mode of file
	chmod(I)	change mode
	chown(I)	change owner
	chown(II)	change owner
	chdir(I)	change working directory
	chdir(II)	change working directory
ascii(VII)	map of ASCII	character set
	getchar(III) read	character
	putchar(III) write	character
		chdir(I) change working directory
		chdir(II) change working directory
check(VIII)	file system consistency	check
		check(VIII) file system consistency check
	chess(VI) the game of	chess
		chess(VI) the game of chess
		chmod(I) change mode
		chmod(II) change mode of file
		chown(I) change owner
		chown(II) change owner
	clri(VIII)	clear i-node
	close(II)	close a file
		close(II) close a file
		clri(VIII) clear i-node
		cmp(I) compare two files
vs(VII)	voice synthesizer	code
	getarg(III) get	command arguments from Fortran
	glob(VII) 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
	gerts(III) Gerts	communication over 201
	dc(IV) DC-11	communications interface
	cmp(I)	compare two files
	proof(I)	compare two text files

	compar(III) default comparison routine for qsort
compar(III) default	comparison routine for qsort
cc(I) C	compiler
tmg(VI)	compiler-compiler
yacc(VI) yet another	compiler-compiler
fc(I) fortran	compiler
wait(I) await	completion of process
cat(I)	concatenate and print
if(I)	conditional command
check(VIII) file system	consistency check
csw(II) read	console switches
kl(IV) KL-11/TTY-33	console typewriter
mkfs(VIII)	construct a file system
ls(I) list	contents of directory
init(VII) process	control initialization
ecvt(III) output	conversion
ctime(III)	convert date and time to ASCII
cp(I)	copy
core(V) format of	core image file
mem(IV)	core memory
	core(V) format of core image file
sin(III) sine,	cosine
nargs(III) argument	count
wc(I) get (English) word	count
tmheader(VII) TM	cover sheet
	cp(I) copy
creat(II)	create a new file
pipe(II)	create a pipe
	creat(II) create a new file
	cref(I) make cross reference listing
cref(I) make	cross reference listing
	crypt(III) password encoding
	csw(II) read console switches
	ctime(III) convert date and time to ASCII
	cubic(VI) three dimensional tic-tac-toe
ttyn(III) return name of	current typewriter
spline(VI) interpolate smooth	curve
dpd(VII) spawn data phone	daemon
	da(IV) voice response unit
dpd(VII) spawn	data phone daemon
dp(IV) dp11 201	data-phone interface
ctime(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
	dc(I) desk calculator
	dc(IV) DC-11 communications interface
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
compar(III)	default comparison routine for qsort
dsw(I)	delete interactively
mesg(I) permit or	deny messages
dup(II) duplicate an open file	descriptor
dc(I)	desk calculator
file(I)	determine format of file
df(VIII)	disk free
wait(II) wait for process to	die
cubic(VI) three	dimensional tic-tac-toe
dir(V) format of	directories
unlink(II) remove	directory entry
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
rf(IV) RF11/RS11 fixed-head	disk file
df(VIII)	disk free
du(I) summarize	disk usage
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
vt(III)	display (vt01) interface
ldiv(III) long	division
dn(IV)	dn11 ACU interface
dn(IV)	dn11 ACU interface
kill(I)	do in an unwanted process
dp(IV)	dp11 201 data-phone interface
dpd(VII)	spawn data phone daemon
dp(IV)	dp11 201 data-phone interface
dsw(I)	delete interactively
du(I)	summarize disk usage
dump(VIII) incremental file system	dump
od(I) octal	dump
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(III)	output conversion
ed(I)	editor
fed(I)	edit associative memory for form letter
a.out(V) assembler and link	editor output
ed(I)	editor
ld(I) link	editor
crypt(III) password	encoding
wc(I) get	(English) word count
nlist(III) get	entries from name list

unlink(II)	remove directory	entry
sky(VI)	obtain	ephemerides
perror(III)	system	error messages
ierror(III)	catch Fortran	errors
pfe(I)	print floating	exception
exec(II)	execute a file	exec(II) execute a file
exec(II)	execute a file	exec(II) execute a file
reset(III)	execute non-local goto	execute non-local goto
sleep(I)	suspend	execution for an interval
sleep(II)	stop	execution for interval
exit(I)	terminate command file	exit(I) terminate command file
exit(II)	terminate process	exit(II) terminate process
exp(III)	exponential function	exp(III) exponential function
exp(III)	exponential function	exp(III) exponential function
pow(III)	floating	exponentiation
greek(VII)	graphics for	extended ascii type-box
factor(VI)	discover prime	factors of a number
factor(VI)	discover prime	factor(VI) discover prime factors of a number
fc(I)	fortran compiler	fc(I) fortran compiler
fed(I)	edit associative memory for form letter	fed(I) edit associative memory for form letter
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
sfs(VI)	structured	file scanner
stat(II)	get	file status
check(VIII)	file system consistency check	check(VIII) file system consistency check
dump(VIII)	incremental	file system dump
restor(VIII)	incremental	file system restore
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
chmod(II)	change mode of	file
close(II)	close a	file
core(V)	format of core image	file
creat(II)	create a new	file
exec(II)	execute a	file
exit(I)	terminate command	file
file(I)	determine format of	file
fstat(II)	get status of open	file
ino(VIII)	get the i-number of a	file(I) determine format of file
link(II)	link to a	file
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

rf(IV) RF11/RS11 fixed-head disk	file
cmp(I) compare two	files
comm(I) print lines common to two	files
size(I) size of an object	file
merge(I) merge several	files
sort(I) sort a	file
proof(I) compare two text	files
reloc(VIII) relocate object	files
rm(I) remove (unlink)	files
sum(I) sum	file
uniq(I) report repeated lines in a	file
write(II) write on a	file
hyphen(VI) find hyphenated words	
typo(I) find possible typos	
rf(IV) RF11/RS11 fixed-head disk	file
pfe(I) print floating exception	
pow(III) floating exponentiation	
fptrap(III) floating point interpreter	
atof(III) ascii to floating	
fork(II) spawn new process	
form(I) form letter generator	
fed(I) edit associative memory for	form letter
core(V) format of core image	file
dir(V) format of directories	
fs(V) format of file system volume	
file(I) determine format of file	
nroff(I) format text	
roff(I) format text	
troff(I) format text	
ar(V) archive (library) file	format
tp(V) DEC/mag tape	formats
printf(III) formatted print	
form(I) form letter generator	
fc(I) fortran compiler	
ierror(III) catch Fortran errors	
setfil(III) specify Fortran file name	
getarg(III) get command arguments from	Fortran
fptrap(III) floating point interpreter	
df(VIII) disk	free
read(II) read	from file
getarg(III) get command arguments	from Fortran
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(III) arc tangent	function
exp(III) exponential	function
sqrt(III) square root	function
bj(VI) the	game of black jack
chess(VI) the	game of chess
moo(VI) guessing	game
m6(VI) general purpose macro processor	
maze(VI) generate a maze problem	

	glob(VII)	generate command arguments
	form(I)	form letter generator
	rand(III)	random number generator
	gerts(III)	Gerts communication over 201
		gerts(III) Gerts communication over 201
	getarg(III)	get command arguments from Fortran
	time(II)	get date and time
	wc(I)	get (English) word count
	nlist(III)	get entries from name list
	stat(II)	get file status
	getgid(II)	get group identification
	getpw(III)	get name from UID
	times(II)	get process times
	fstat(II)	get status of open file
	ino(VIII)	get the i-number of a file
	tty(I)	get typewriter name
	gtty(II)	get typewriter status
	getuid(II)	get user identification
		getarg(III) get command arguments from Fortran
		getchar(III) read character
		getc(III) buffered input
		getgid(II) get group identification
		getpw(III) get name from UID
		getty(VII) set typewriter mode
		getuid(II) get user identification
		glob(VII) generate command arguments
		goto(I) command transfer
reset(III)		execute non-local goto
	greek(VII)	graphics for extended ascii type-box
	plot(I)	make a graph
		greek(VII) graphics for extended ascii type-box
		grep(I) search a file for a pattern
	getgid(II)	get group identification
	setgid(II)	set process group ID
		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
	wump(VI)	hunt the wumpus
	hyphen(VI)	find hyphenated words
		hyphen(VI) find hyphenated words
	hypot(III)	calculate hypotenuse
		hypot(III) calculate hypotenuse
	getgid(II)	get group identification
	getuid(II)	get user identification
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
	kill(I) do	in an unwanted process
	dump(VIII)	incremental file system dump
	restor(VIII)	incremental file system restore
ptx(VI)	permuted	index
	indir(II)	indirect system call
	indir(II)	indirect system call
	utmp(V) user	information
init(VII)	process control	initialization
	init(VII)	process control initialization
	clri(VIII) clear	i-node
	ino(VIII)	get the i-number of a file
getc(III)	buffered	input
	20boot(VIII)	install new 11/20 system
	dsw(I) delete	interactively
	tss(I)	interface to MH-TSS
	cat(IV) phototypesetter	interface
dc(IV)	DC-11 communications	interface
	dn(IV) dn11 ACU	interface
	dp(IV) dp11 201 data-phone	interface
	tiu(IV) Spider	interface
tm(IV)	TM-11/TU-10 magtape	interface
	vs(IV) voice synthesizer	interface
	vt(III) display (vt01)	interface
	vt(IV) 11/20 (vt01)	interface
	spline(VI)	interpolate smooth curve
	fptrap(III) floating point	interpreter
	sh(I) shell (command)	interpreter
	sno(I) Snobol	interpreter
sleep(I)	suspend execution for an	interval
	sleep(II) stop execution for	interval
	split(I) split a file	into pieces
	ino(VIII) get the	i-number of a file
	bj(VI) the game of black	jack
	kill(I) do	in an unwanted process
	kill(II)	send signal to a process
	kl(IV)	KL-11/TTY-33 console typewriter
	kl(IV)	KL-11/TTY-33 console typewriter
	ld(I)	link editor
	ldiv(III)	long division
	form(I) form	letter generator
fed(I)	edit associative memory for form	letter
	ar(V) archive	(library) file format
	ar(I) archive and	library maintainer
	opr(I) off	line print
	comm(I) print	lines common to two files
	uniq(I) report repeated	lines in 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
log(III)	natural	logarithm
		log(III) natural logarithm
wtmp(V)	user	login history
passwd(I)	set	login password
		login(I) sign onto UNIX
	ldiv(III)	long division
nice(I)	run a command at	low priority
		ls(I) list contents of directory
		m6(VI) general purpose macro processor
m6(VI)	general purpose	macro processor
tm(IV)	TM-11/TU-10	magtape interface
tp(I)	manipulate DECtape and	magtape
	mail(I) send	mail to another user
		mail(I) send mail to another user
ar(I)	archive and library	maintainer
	mknod(II)	make a directory or a special file
	mkdir(I)	make a directory
	plot(I)	make a graph
	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
man(I)	run off section of UNIX	manual
	ascii(VII)	map of ASCII character set
maze(VI)	generate a	maze problem
		maze(VI) generate a maze problem
		mem(IV) core memory
fed(I)	edit associative	memory for form letter
	mem(IV) core	memory
	merge(I)	merge several files
		merge(I) merge several files
		mesg(I) permit or deny messages
		mesg(III) write message on typewriter
	mesg(III) write	message on typewriter
mesg(I)	permit or deny	messages
perror(III)	system error	messages
tss(I)	interface to	MH-TSS
	msh(VII)	mini-shell
		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(VII)	set typewriter	mode
		moo(VI) guessing game
	mount(II)	mount file system

mount(VIII)	mount file system
	mount(II) mount file system
	mount(VIII) mount file system
mv(I)	move or rename a file
seek(II)	move read/write pointer
rp(IV) RP-11/RP03	moving-head disk
	msh(VII) mini-shell
	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
setfil(III) specify Fortran file	name
tty(I) get typewriter	name
	nargs(III) argument count
log(III)	natural logarithm
20boot(VIII) install	new 11/20 system
creat(II) create a	new file
fork(II) spawn	new process
	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(III) execute	non-local goto
	nroff(I) format text
rand(III) random	number generator
factor(VI) discover prime factors of a	number
size(I) size of an	object file
reloc(VIII) relocate	object files
sky(VI)	obtain ephemerides
azel(VI)	obtain satellite predictions
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
	opr(I) off line print
stty(I) set teletype	options
rk(IV) RK-11/RK03	(or RK05) disk
ecvt(III)	output conversion
a.out(V) assembler and link editor	output
putc(III) buffered	output
gerts(III) Gerts communication	over 201
ov(VI)	overlay pages
	ov(VI) overlay pages
chown(I) change	owner
chown(II) change	owner
ov(VI) overlay	pages

pc(IV) PC-11 paper tape reader/punch  
 passwd(I) set login password  
 passwd(V) password file  
 crypt(III) password encoding  
 passwd(V) password file  
 passwd(I) set 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  
 ptx(VI) permuted index  
 perror(III) system error messages  
 pfe(I) print floating exception  
 dpd(VII) spawn data phone daemon  
 cat(IV) phototypesetter interface  
 catsim(I) phototypesetter simulator  
 split(I) split a file into pieces  
 pipe(II) create a pipe  
 pipe(II) create a pipe  
 plot(I) make a graph  
 fptrap(III) floating point interpreter  
 seek(II) move read/write pointer  
 typo(I) find possible typos  
 pow(III) floating exponentiation  
 azel(VI) obtain satellite predictions  
 pr(I) print file  
 factor(VI) discover prime factors of a number  
 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  
 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  
 maze(VI) generate a maze problem  
 boot procedures(VIII) UNIX startup  
 init(VII) process control initialization  
 setgid(II) set process group ID  
 ps(I) process status  
 times(II) get process times  
 wait(II) wait for process to die  
 setuid(II) set process user ID  
 exit(II) terminate process  
 fork(II) spawn new process  
 kill(I) do in an unwanted process  
 kill(II) send signal to a process



m6(VI) general purpose macro	processor
wait(I) await completion of	process
hmul(III) high-order	product
break(II) set	program break
nice(II) set	program priority
	proof(I) compare two text files
	ps(I) process status
	ptx(VI) permuted index
m6(VI) general	purpose macro processor
	putchar(III) write character
	putc(III) buffered output
compar(III) default comparison routine for	qsort
	qsort(III) quicker sort
qsort(III) quicker sort	rand(III) random number generator
	rand(III) random number generator
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
reloc(VIII)	relocate object files
strip(I) remove symbols and	relocation bits
	reloc(VIII) relocate object files
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(III) execute non-local goto
da(IV) voice	response unit
restor(VIII) incremental file system	restore
	restor(VIII) incremental file system restore
ttyn(III)	return name of current typewriter
	rew(I) rewind tape
rew(I)	rewind tape
rf(IV)	RF11/RS11 fixed-head disk file
	rf(IV) RF11/RS11 fixed-head disk file
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
compar(III) default comparison	routine for qsort
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  
 azel(VI) obtain satellite predictions  
 sfs(VI) structured file scanner  
     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 another user  
     kill(II) send signal to a process  
 passwd(I) set login password  
     stty(II) set mode of typewriter  
     setgid(II) set process group ID  
     setuid(II) set process user ID  
     break(II) set program break  
     nice(II) set program priority  
     tabs(VII) set tab stops  
     stty(I) set teletype options  
 date(I) print and set the date  
     stime(II) set time  
     getty(VII) set typewriter mode  
 ascii(VII) map of ASCII character set  
     setfil(III) specify Fortran file name  
     setgid(II) set process group ID  
     setuid(II) set process user ID  
 merge(I) merge several files  
     sfs(VI) structured file scanner  
 tmheader(VII) TM cover sheet  
     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  
 catsim(I) phototypesetter simulator  
     sin(III) sine, cosine  
         sin(III) sine, cosine  
     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(I) Snobol interpreter  
         sno(I) Snobol interpreter  
     sort(I) sort a file  
         sort(I) sort a file  
 qsort(III) quicker sort  
     dpd(VII) spawn data phone daemon  
     fork(II) spawn new process  
     speak(I) word to voice translator

mknod(II)	make a directory or a	special file
mknod(VIII)	build	special file
setfil(III)		specify Fortran file name
tiu(IV)		Spider interface
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
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(VII)	set tab	stops
strip(I)		remove symbols and relocation bits
sfs(VI)		structured file scanner
stty(I)		set teletype options
stty(II)		set mode of typewriter
sum(I)		sum file
sum(I)		sum file
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
switch(III)		switch on value
csw(II)	read console	switches
switch(III)		switch on value
strip(I)	remove	symbols and relocation bits
sync(II)		update super-block
sync(VIII)		update the super block
vs(VII)	voice	synthesizer code
vs(IV)	voice	synthesizer interface
indir(II)	indirect	system call
check(VIII)	file	system consistency check
dump(VIII)	incremental file	system dump
perror(III)		system error messages
restor(VIII)	incremental file	system restore
fs(V)	format of file	system volume
20boot(VIII)	install new 11/20	system
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(VII)	set	tab stops
tabs(VII)	set	tab stops

atan(III) arc	tangent function
tp(V) DEC/mag	tape formats
pc(IV) PC-11 paper	tape reader/punch
rew(I) rewind	tape
tc(IV)	TC-11/TU56 DECTape
tc(IV)	TC-11/TU56 DECTape
stty(I) set	teletype options
exit(I)	terminate command file
exit(II)	terminate process
proof(I) compare two	text files
nroff(I) format	text
roff(I) format	text
troff(I) format	text
cubic(VI)	three dimensional tic-tac-toe
cubic(VI) three dimensional	tic-tac-toe
ttt(VI)	tic-tac-toe
time(I)	time a command
ctime(III) convert date and	time to ASCII
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
tiu(IV)	Spider interface
tmheader(VII)	TM cover sheet
tm(IV)	TM-11/TU-10 magtape interface
tmg(VI)	compiler-compiler
tmheader(VII)	TM cover sheet
tm(IV)	TM-11/TU-10 magtape interface
tp(I)	manipulate DECTape and magtape
tp(V)	DEC/mag tape formats
goto(I) command	transfer
speak(I) word to voice	translator
tr(I)	transliterate
tr(I)	transliterate
troff(I)	format text
tss(I)	interface to MH-TSS
ttt(VI)	tic-tac-toe
tty(I)	get typewriter name
ttyn(III)	return name of current typewriter
cmp(I) compare	two files
comm(I) print lines common to	two files
proof(I) compare	two text files
type(I)	type on 2741
greek(VII) graphics for extended ascii	type-box
type(I)	type on 2741
getty(VII) set	typewriter mode
tty(I) get	typewriter name
gtty(II) get	typewriter status
kl(IV) KL-11/TTY-33 console	typewriter
mesg(III) write message on	typewriter
stty(II) set mode of	typewriter

tty(III) return name of current	typewriter
typo(I) find possible	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
da(IV) voice response	unit
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
kill(I) do in an	unwanted process
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 identification
setuid(II) set process	user ID
utmp(V)	user information
wtmp(V)	user login history
mail(I) send mail to another	user
su(VIII) become privileged	user
write(I) write to another	user
	utmp(V) user information
switch(III) switch on	value
da(IV)	voice response unit
vs(VII)	voice synthesizer code
vs(IV)	voice synthesizer interface
speak(I) word to	voice translator
fs(V) format of file system	volume
	vs(IV) voice synthesizer interface
	vs(VII) voice synthesizer code
vt(III) display	(vt01) interface
vt(IV) 11/20	(vt01) interface
	vt(III) display (vt01) interface
	vt(IV) 11/20 (vt01) interface
wait(II)	wait for process to die
	wait(I) await completion of process
	wait(II) wait for process to die
	wc(I) get (English) word count
who(I)	who is on the system
	who(I) who is on the system
wc(I) get (English)	word count
speak(I)	word to voice translator
hyphen(VI) find hyphenated	words
chdir(I) change	working directory
chdir(II) change	working directory
putchar(III)	write character
mesg(III)	write message on typewriter
write(II)	write on a file

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) hunt the wumpus  
wump(VI) hunt the wumpus  
yacc(VI) yet another compiler-compiler  
yacc(VI) yet another compiler-compiler