PA-1C600-01
Section 2
Issue 1, October 1977
AT&TCo SPCS

UNIX PROGRAMMER'S MANUAL
of MERT Release 0

Based on Program Generic PG-1C300 Issue 3
for inclusion in PG-1C600 Issue 1
with updated pages

Published by Department 8234

March 1977
(Updated October 1977)

In Memoriam Joseph F. Ossanna

Creator of the UNIX Text Formatting Program *troff*

This manual was photocomposed in the Murray Hill Computation Center. The text of the manual was prepared using the UNIX* *ed* text editor and *troff* formatting program, as well as a *Stare* graphic hardcopy device for assistance in the proof correction process.

* Trademark of Bell Laboratories

Second Printing

Reproduction, assembly and distribution:
Technical Documentation Department
Bell Laboratories, Whippany, New Jersey

PREFACE
to the version included with MERT Release 0

This version of the UNIX PROGRAMMER'S MANUAL is essentially the Program Generic 3 Edition (March 1977) with update pages to reflect the UNIX commands and system calls distributed with and supported under MERT Release 0. If the update pages have not yet been included into the UNIX PROGRAMMER'S MANUAL Section of the MERT Release 0 Manual, you should do so, replacing the superseded pages. A list of update pages with instructions can be found on the next page.

Please send suggestions and corrections concerning this manual to Mrs. R. J. Fiore, Murray Hill, room 2F-219.

G.W.R.L.
October 1977

PREFACE
to the Generic 3 Edition

This document is published as part of the UNIX Operating System Program Generic, PG-1C300 Issue 3. The development of the Program Generic is the result of the efforts of the members of the Small Systems Development Department (8234).

Most of the commands and system software were written by the Computing Science Research Center (127), especially K. Thompson and D. M. Ritchie. This manual is based on the UNIX PROGRAMMER'S MANUAL, Sixth Edition, May, 1975 by K. Thompson and D. M. Ritchie.

For corrections and comments please contact I. A. Hahner, MH 2F-219, Extension 2771.

J. F. M.
March 1977

List of Update Pages for Unix Generic 3 in MERT Release 0 Manual

In the following, bracketed [] names of replacing pages are used to denote essentially unchanged pages, which are being replaced because of two-sided printing. New or changed pages are enclosed in braces {} and printed in bold face.

INTRODUCTION

Replace page 1-45 (cover page through index) by new section, i.e. everything in front of "I Commands" divider.

I COMMANDS

Replace [adb] with {**adb**}
After [bas] insert {**basename**}
Replace [cat] through [chdir] with [cat] {**cc**} [chdir]
Replace [cref] [date] with [cref] {**cpio**} {**crypt**} {**date**} {**dirname**}
Replace [echo] [ed] [eqn] [exit]
 with {**echo**} {**ed**} [eqn] [exit]
Replace [goto] [grep] [help] [if] [kill] [lc] with [grep] [help] [kill]
Replace [line] [ln] with {**lint**} [ln]
Replace [ls] [mail] with {**ls**} [mail]
Insert {**make**} {**man**} before [mesg]
Insert {**newgrp**} between [neqn] and [nice]
Replace [nohup] [nroff] with [nohup] {**nroff**}
Replace [od] [onintr] [passwd] [pfe] with {**od**} {**passwd**}
Replace [prt] [ps] [pwd] with [prt] {**ps**} [pwd]
Replace [read] [return] with {**read**} [return]
Replace [rmdir] [sed] [sh] [shift] [size] sleep] with
 [rmdir] [sed] {**sh**} {**shift**} [size] {**sleep**}
Replace [size] [sleep] with [size] {**sleep**}
Replace [tee] [time] with [tee] {**test**} [time]
Replace [tr] [troff] [typo] with [tr] {**troff**} {**tty**} [typo]
Replace [write] [yacc] with [write] {**yacc**}

II SYSTEM CALLS

Replace [Intro] [access] [acct] with {**Intro**} [access]
Replace [call} with {**call**}
Replace [chown] [chroot] with [chown]
Replace [errlog] [exec] [exit] [fork] [fstat]
 with [exec] [exit] [fork] {**fstat**}
Replace [getgid] [getpid] [getuid] [gtty] with {**getgid**} {**getpid**} {**getuid**} [gtty]
Replace [indir] [ioctl] with [indir]
Insert {**loginfo**}
Replace [profile] [ptrace] [read] with [profile] {**read**}
Replace [seek] [setgid] with {**seek**} [setgid]
Replace [stat] with {**stat**}
Replace [stime] [stty] [sync] with [stime] {**stty**} [sync]
Replace [unlink] [wait] [write] with [unlink] {**wait**} [write]

III SUBROUTINES
Insert {**Intro**} before [abort]
Replace [log] [lseek] with [log] {**lseek**}
Replace [monitor] [nargs] with [monitor] {**newio**}

IV DRIVERS
Replace [cm] [dc] with {**Intro**} [dc]
Replace [dh] [dn] [dp] [hd]
 with [dh] {**dm**} {**dmc**} [dn] [dp] {**dr**} {**du**}
Replace [mem] [pc] with [mem]
Replace [rp] [tc] with [rp] {**sdh**} [tc] {**tf**}
Replace [tm] [tty] with [tm] {**tty**}

V FILE FORMATS
Replace [a.out] [acct] [archive] [core] [directory] [dump]
 with {**Intro**} [a.out] [archive] [core] {**cpio**} [directory] {**dump**}
Replace [fs] [lines] [passwd] [sccsfile] [tp]
 with {**fs**} {**include**} {**man**} [passwd] [sccsfile] [tp] {**ttys**}

VI USER PROGRAMS
In the past, inclusion of commands in Section VI rather than Section I has implied a lower level of support. Commands which have proven to be valuable and much used have been moved from Section VI to Section I. With this release, this practice has not been followed to reduce the number of pages to be reprinted just for the reason of getting a new section number. Thus, commands like *lex* and *tbl* and others can be expected to move into Section I in the next release.

Replace [cal] [chess] with [cal] {**cb**} [chess] {**col**}
Replace [cubic] [factor] [fed] [form]
 with [cubic] {**cut**} {**db**} {**deroff**} {**egrep**} {**fgrep**}
Insert {**join**} between [hyphen] and [lex]
Replace [moo] [ptx] [reform] [sno]
 with {**lint**} {**m4**} [moo] {**paste**} [ptx] {**rc**} [reform] {**rev**} [sno] {**spell**} {**spline**} {**tabs**}
Replace [ttt] [wump] with [ttt] {**units**} {**uucp**} [wump]

VIII SYSTEM PROGRAMS
Replace all of Section VIII (10). This replacement corresponds to the following changes:

Replace [ac] [accton] [boot] [check] [chown]
 with {**Intro**} [ac] {**boot**} {**check**} [chown]
Replace [dcheck] [df] [dump] [errdemon] [errpt] [getty]
 with {**dcheck**} {**df**} {**dump**} {**fsck**} {**getty**}
Replace [glob] [icheck] [init] [ino] [iostat] [load] [lpd]
 with [glob] {**icheck**} {**init**} [lpd]
Remove [mkconf]
Replace [mknod] [mkpt] [mount]
 with {**mknod**} {**mkpt**} [mount]
Replace [restor] [sa] with {**restor**}
Replace [telinit] [umount] with [umount]

SECTION RENUMBERING IN UNIX PROGRAMMER'S MANUAL

Since a modified but not reprinted version of 1C-300 has been incorporated in PA 1C600 the sections of 1C300 had to be renumbered. The new numbers will, of course, not appear on the old (i.e. not reprinted) pages of 1C300. The following mapping applies:

New Section Number	*Old Section Number*
1C600 Section 2	1C300 Section 1 before Commands divider
1C600 Section 3	1C300 Section 1 behind Commands Divider (I)
1C600 Section 4	1C300 Section 2 (II)
1C600 Section 5	1C300 Section 3 (III)
1C600 Section 6	1C300 Section 4 (IV)
1C600 Section 7	1C400 Section 5 (V)
1C600 Section 8	1C400 Section 6 (VI)
1C600 Section 9	1C400 Section 7 (VII)
1C600 Section 10	1C400 Section 8 (VIII)

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, the 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.	Drivers
V.	File Formats
VI.	User Programs
VII.	Tables
VIII.	System Programs

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.

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

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 Programs (section VI), while part of the Standard UNIX system, are not fully supported, and the principal reason for listing them is to indicate their existence without necessarily giving a complete description.

Section VII groups together the information pertaining to tabular data.

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 one or more pages. Below the program application heading is the name of the entry in bold-face type. Entries within each section are alphabetized. The page numbers of each entry start at 1.

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.

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 (PA-1C3019).

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 subdirectory (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 K. Thompson and D. M. Ritchie (PD-1C300 Section1). 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, 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.

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

ltod	double precision integer to floating point conversion
ltoi	long integer to integer conversion
mesg	write message on typewriter
mktemp	make a unique named temporary file
monitor	prepare execution profile
newio	a new IO subroutine package
fopen	open file
freopen	reopen file
getc	get character
fgetc	get character
putc	put character
fputc	put character
fclose	close file
fflush	flush buffer
exit	exit from subroutine
feof	end-of-file
ferror	error exit
getchar	get character
putchar	put character
gets	get string
fgets	get string
puts	put string
fputs	put string
ungetc	push character back
printf	print formatted
fprintf	print formatted
sprintf	print formatted
scanf	input conversion
fscanf	input conversion
sscanf	input conversion
fread	read from file
fwrite	write to file
rewind	rewind
system	execute command
atof	ASCII to float conversion
tmpnam	create tmp name
abort	abort process
intss	test for tss or batch
cfree	deallocate memory
wdleng	find machine word size
calloc	allocate memory
getw	get word
putw	put word
setbuf	set buffer size
fileno	get file descriptor
fseek	seek to offset
ftell	get current offset
getpw	get password line
strcat	concatenate strings
strcmp	compare strings
strcpy	copy string
strlen	obtain string length
isalpha	test for alphabetic
isupper	test for upper case
islower	test for lower case
isdigit	test for numeric
isspace	test for space
toupper	translate to upper case
tolower	translate to lower case
nlist	get entries from name list
perror, sys_errlist, sys_nerr, errno	system error messages
pow	floating exponentiation
printf	formatted print
putc, putw, fcreat, fflush	buffered output
putchar, flush	write character
qsort	quicker sort

PERMUTED INDEX

dp(IV) DP-11	201 data-phone interface
diff3(I)	3-way differential file comparison
	abort in newio(III) abort process
abort in newio(III)	abort process
	abort(III) generate an IOT fault
	abs, fabs(III) absolute value
abs, fabs(III)	absolute value
access(II) determine	accessibility of file
	access(II) determine accessibility of file
acct(V)	Accounting file
ac(VIII) login	accounting
	acct(V) Accounting file
alarm(II)	activate alarm clock timer
dn(IV) DN-11	ACU interface
	ac(VIII) login accounting
	adb(I) debugger
shift(I)	adjust Shell arguments
	admin(I) administer SCCS files
admin(I)	administer SCCS files
basename(I) strip filename	affixes
	agen(VI) generate associative memory drivers
alarm(II) activate	alarm clock timer
	alarm(II) activate alarm clock timer
calloc in newio(III)	allocate memory
break, brk, sbrk(II) change core	allocation
alloc(III) core	allocator
	alloc(III) core allocator
isalpha in newio(III) test for	alphabetic
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
cpio(V) format of cpio	archive
cpio(I) copy file	archives in and out
getarg, iargc(III) get command	arguments from Fortran
echo(I) echo	arguments
eval in sh(I) evaluate	arguments
glob(VIII) 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 in newio(III)	ASCII to float conversion
atof(III) convert	ASCII to floating
atoi(III) convert	ASCII to integer
gmtime(III) convert date and time to	ASCII...ctime, localtime,
	ascii(VII) map of ASCII character set
	as(I) assembler
help(I)	ask for help
a.out(V)	assembler and link editor output
as(I)	assembler
agen(VI) generate	associative memory drivers
dm(IV)	asynchronous communication device
tty(IV) interface to low speed	asynchronous devices including typewriters
kl(IV) KL-11 or DL-11	asynchronous interface
	atan, atan2(III) arc tangent function
atan,	atan2(III) arc tangent function
	atof in newio(III) ASCII to float conversion
	atof(III) convert ASCII to floating
	atoi(III) convert ASCII to integer

wait(I)	await completion of process
ungetc in newio(III) push character	back
join(VI) relational data	base operator
	basename(I) strip filename affixes
	bas(I) basic
bas(I)	basic
intss in newio(III) test for tss or	batch
	bc(I) arbitrary précision interactive language
cb(VI) C	beautifier
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) MERT startup
	break, brk, sbrk(II) change core allocation
	break in sh(I) exit from loop
break,	brk, sbrk(II) change core allocation
setbuf in newio(III) set	buffer size
getc, getw, fopen(III)	buffered input
putc, putw, fcreat, fflush(III)	buffered output
fflush in newio(III) flush	buffer
mknod(VIII)	build special file
cb(VI)	C beautifier
cc(I)	C compiler
lint(I) a	C program verifier
hypot(III)	calculate hypotenuse
dc(I) desk	calculator
cal(VI) print	calendar
	call, lcall, vcall(II) create and execute a new process
indir(II) indirect system	call
	calloc in newio(III) allocate memory
	cal(VI) print calendar
islower in newio(III) test for lower	case
isupper in newio(III) test for upper	case
in newio(III) translate to lower	case...tolower
in newio(III) translate to upper	case...toupper
ierror(III)	catch Fortran errors
signal(II)	catch or ignore signals
trap in sh(I)	catch signals
	cat(I) concatenate and print
	cb(VI) C beautifier
	cc(I) C compiler
chdir,	cd(I) change working directory
floor,	ceil(III) floor and ceiling functions
floor, ceil(III) floor and	ceiling functions
	cfree in newio(III) deallocate memory
break, brk, sbrk(II)	change core allocation
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
chroot(I)	change root directory for a command
chdir, cd(I)	change working directory
chdir(II)	change working directory
pipe(II) create an interprocess	channel
ungetc in newio(III) push	character back
gsi(VI) interpret extended	character set on GSI terminal
ascii(VII) map of ASCII	character set
fgetc in newio(III) get	character
fputc in newio(III) put	character
getc in newio(III) get	character
getchar in newio(III) get	character
getchar(III) read	character

putc in newio(III) put	character
putchar, flush(III) write	character
putchar in newio(III) put	character
	chdir, cd(I) change working directory
	chdir(II) change working directory
fsck(VIII) file system consistency	check and interactive repair
check(VIII) file system consistency	check
file system directory consistency	check...dcheck(VIII)
file system storage consistency	check...icheck(VIII)
	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(II) change owner and group of a file
	chown(VIII) change owner
	chroot(I) change root directory for a command
clri(VIII)	clear i-node
cron(VIII)	clock daemon
alarm(II) activate alarm	clock timer
close(II)	close a file
fclose in newio(III)	close 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 programming language
chroot(I) change root directory for a	command
system in newio(III) execute	command
test(I) condition	command
time(I) time a	command
	comm(I) print lines common to two files
comm(I) print lines	common to two files
dm(IV) asynchronous	communication device
du(IV) DU-11 synchronous	communication device
dc(IV) DC-11	communications interface
dh(IV) DH-11	communications multiplexer
diff(I) differential file	comparator
strcmp in newio(III)	compare strings
cmp(I)	compare two files
	compar(III) default comparison routine for qsort
compar(III) default	comparison routine for qsort
diff3(I) 3-way differential file	comparison
cc(I) C	compiler
yacc(I) yet another	compiler-compiler
fc(I) Fortran	compiler
rc(VI) Ratfor	compiler
wait(I) await	completion of process
cat(I)	concatenate and print
strcat in newio(III)	concatenate strings
test(I)	condition command
fsck(VIII) file system	consistency check and interactive repair
check(VIII) file system	consistency check
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
deroff(VI) remove Troff and Eqn	constructs
egrep(VI) search a file for lines	containing a pattern
fgrep(VI) search a file for lines	containing keywords
ls(I) list	contents of directory

	continue in sh(I) next iteration in loop
init(VIII) process	control initialization
units(VI)	conversion program
atof in newio(III) ASCII to float	conversion
floating point to double precision integer	conversion...dtol(III)
ecvt, fcvt(III) output	conversion
fscanf in newio(III) input	conversion
itol(III) integer to long integer	conversion
locv(III) long output	conversion
double precision integer to floating point	conversion...ltod(III)
ltoi(III) long integer to integer	conversion
scanf in newio(III) input	conversion
sscanf in newio(III) input	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
cpall(I)	copy all files to a directory
cpio(I)	copy file archives in and out
strcpy in newio(III)	copy string
cp(I)	copy
uucp(VI) unix-to-unix	copy
break, brk, sbrk(II) change	core allocation
alloc(III)	core allocator
core(V) format of	core image file
mem, kmem, null(IV)	core memory
	core(V) format of core image file
sin,	cos(III) trigonometric functions
wc(I) word	count
	cpall(I) copy all files to a directory
	cp(I) copy
cpio(V) format of	cpio archive
	cpio(I) copy file archives in and out
	cpio(V) format of cpio archive
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
call, lcall, vcall(II)	create and execute a new process
tmpnam in newio(III)	create tmp name
	creat(II) create a new file
	cref(I) make cross reference listing
	cron(VIII) clock daemon
cref(I) make	cross reference listing
	crypt(I) encode/decode
	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
ftell in newio(III) get	current offset
lnxx(III) return name of	current terminal
spline(VI) interpolate smooth	curve
cut(VI)	cut out selected fields of each line of a file
	cut(VI) cut out selected fields of each line of a file
cron(VIII) clock	daemon
lpd(VIII) line printer	daemon
dp(IV) DP-11 201	data-phone interface
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(VI) debug
dc(IV)	DC-11 communications interface
	dcheck(VIII) file system directory consistency check
	dc(I) desk calculator

	dc(IV) DC-11 communications interface
dmc(IV) network link with	DDCMP protocol
	dd(I) convert and copy a file
cfree in newio(III)	deallocate memory
db(VI)	debug
adb(I)	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
include(V) system data structure	definitions file
dsw(I)	delete interactively
tail(I)	deliver the last part of a file
delta(I) make an SCCS	delta
	delta(I) make an SCCS delta
mesg(I) permit or	deny messages
	deroff(VI) remove Troff and Eqn constructs
dup(II) duplicate an open file	descriptor
fileno in newio(III) get file	descriptor
mail(I) send mail to	designated users
dc(I)	desk calculator
access(II)	determine accessibility of file
file(I)	determine format of file
dr(IV) DR-11 general	device interface
tty(IV) interface to low speed asynchronous	devices including typewriters
	df(VIII) disk free
dh(IV)	DH-11 communications multiplexer
sdh(IV)	DH11 for Satellite Processor System
	dh(IV) DH-11 communications multiplexer
	diff3(I) 3-way differential file comparison
diff(I)	differential file comparator
diff3(I) 3-way	differential file comparison
	diff(I) differential file comparator
cubic(VI) three	dimensional tic-tac-toe
loginfo(II) login inform.: name,	dir, tty, post; udata
dir(V) format of	directories
dcheck(VIII) file system	directory consistency check
unlink(II) remove	directory entry
chroot(I) change root	directory for a command
pwd(I) working	directory name
mknod(II) make a	directory or a special file
chdir, cd(I) change working	directory
chdir(II) change working	directory
cpall(I) copy all files to a	directory
ls(I) list contents of	directory
mkdir(I) make a	directory
mvall(I) move all files to a	directory
rmdir(I) remove	directory
	dirname(I) strip simple filename
	dir(V) format of directories
tf(IV) Telefile	disk driver
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
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
kl(IV) KL-11 or	DL-11 asynchronous interface
	dmc(IV) network link with DDCMP protocol
	dm(IV) asynchronous communication device
dn(IV)	DN-11 ACU interface
	dn(IV) DN-11 ACU interface
man(I) print on-line	documentation

dtol(III) floating point to	double precision integer conversion
ltod(III)	double precision integer to floating point conversion
dp(IV)	DP-11 201 data-phone interface
	dp(IV) DP-11 201 data-phone interface
dr(IV)	DR-11 general device interface
	dr(IV) DR-11 general device interface
agen(VI) generate associative memory	drivers
Intro(IV) INTROD. TO	DRIVERS
tf(IV) Telefile disk	driver
	dsw(I) delete interactively
conversion...	dtol(III) floating point to double precision integer
du(IV)	DU-11 synchronous communication device
	du(I) summarize disk usage
	du(IV) DU-11 synchronous communication device
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
cut(VI) cut out selected fields of	each line of a file
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
a.out(V) assembler and link	editor output
ed(I) text	editor
ld(I) link	editor
sed(I) stream	editor
	egrep(VI) search a file for lines containing a pattern
crypt(I)	encode/decode
crypt(III) password	encoding
	end, etext, edata(III) last locations in program
feof in newio(III)	end-of-file
nlist(III) get	entries from name list
unlink(II) remove directory	entry
deroff(VI) remove Troff and	Eqn constructs
	eqn(I) typeset mathematics
perror, sys_errlist, sys_nerr,	errno(III) system error messages
ferror in newio(III)	error exit
sys_nerr, errno(III) system	error messages...perror, sys_errlist,
ierror(III) catch Fortran	errors
spell(VI) find spelling	errors
end,	etext, edata(III) last locations in program
	eval in sh(I) evaluate arguments
eval in sh(I)	evaluate arguments
	exec, execl, execv(II) execute a file
	exec in sh(I) execute within shell
exec,	execl, execv(II) execute a file
exec, execl, execv(II)	execute a file
call, lcall, vcall(II) create and	execute a new process
system in newio(III)	execute command
reset, setexit(III)	execute non-local goto
exec in sh(I)	execute within shell
sleep(I) suspend	execution for an interval
sleep(II) stop	execution for interval
pause(II) suspend	execution indefinitely
monitor(III) prepare	execution profile
profil(II)	execution time profile
exec, execl,	execv(II) execute a file
break in sh(I)	exit from loop
exit in newio(III)	exit from subroutine
	exit in newio(III) exit from subroutine
ferror in newio(III) error	exit

	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(VII) graphics for	extended TTY-37 type-box
abs,	fabs(III) absolute value
abort(III) generate an IOT	fault
	fc(I) Fortran compiler
	fclose in newio(III) close file
putc, putw,	fcreat, fflush(III) buffered output
ecvt,	fcvt(III) output conversion
col(VI) filter reverse line	feeds
	feof in newio(III) end-of-file
	ferror in newio(III) error exit
	fflush in newio(III) flush buffer
putc, putw, fcreat,	fflush(III) buffered output
	fgetc in newio(III) get character
	fgets in newio(III) get string
	fgrep(VI) search a file for lines containing keywords
cut(VI) cut out selected	fields of each line of a file
cpio(I) copy	file archives in and out
diff(I) differential	file comparator
diff3(I) 3-way differential	file comparison
dup(II) duplicate an open	file descriptor
fileno in newio(III) get	file descriptor
grep(I) search a	file for a pattern
egrep(VI) search a	file for lines containing a pattern
fgrep(VI) search a	file for lines containing keywords
mkpt(VIII) make prototype	file for use by mkfs
ar(V) archive (library)	file format
Intro(II) INTROD. TO MERT	FILE FORMATS
Intro(V) INTROD. TO	FILE FORMATS
split(I) split a	file into pieces
setfil(III) specify Fortran	file name
tell(II) get	file offset
stat(II) get	file status
fsck(VIII)	file system consistency check and interactive repair
check(VIII)	file system consistency check
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(VII) mounted	file system table
fs(V) format of UNIX	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
cut out selected fields of each line of a	file...cut(VI)
fclose in newio(III) close	file
fopen in newio(III) open	file
fread in newio(III) read from	file
freopen in newio(III) reopen	file
fwrite in newio(III) write to	file
	file(I) determine format of file
system data structure definitions	file...include(V)
basename(I) strip	filename affixes
dirname(I) strip simple	filename
	fileno in newio(III) get file descriptor
cpall(I) copy all	files to a directory
mvall(I) move all	files to a directory
col(VI)	filter reverse line feeds
find(I)	find files

hyphen(VI)	find hyphenated words
wdleng in newio(III)	find machine word size
typo(I)	find possible typos
spell(VI)	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
atof in newio(III) ASCII to	float conversion
pow(III)	floating exponentiation
fmod(III)	floating modulo function
ltod(III) double precision integer to	floating point conversion
fptrap(III)	floating point interpreter
dtol(III)	floating point to double precision integer conversion
atof(III) convert ASCII to	floating
floor, ceil(III)	floor and ceiling functions
	floor, ceil(III) floor and ceiling functions
fflush in newio(III)	flush buffer
putchar,	flush(III) write character
	fmod(III) floating modulo function
	fopen in newio(III) open file
getc, getw,	fopen(III) buffered input
	fork(II) spawn new process
core(V)	format of core image file
cpio(V)	format of cpio archive
dir(V)	format of directories
file(I) determine	format of file
sccsfile(V)	format of SCCS file
fs(V)	format of UNIX file system volume
tbl(VI)	format tables for nroff or troff
ar(V) archive (library) file	format
dump(V) incremental dump tape	format
man(V) manual page	format
Intro(II) INTROD. TO MERT FILE	FORMATS
Intro(V) INTROD. TO FILE	FORMATS
tp(V) DEC/mag tape	formats
printf(III)	formatted print
fprintf in newio(III) print	formatted
printf in newio(III) print	formatted
sprintf in newio(III) print	formatted
nroff, troff(I) text	formatters
nroff, troff(I) text	formatters
tmac(VI) macros for	formatting manuscripts
fc(I)	Fortran compiler
ierror(III) catch	Fortran errors
setfil(III) specify	Fortran file name
iargc(III) get command arguments from	Fortran...getarg,
	fprintf in newio(III) print formatted
	fptrap(III) floating point interpreter
	fputc in newio(III) put character
	fputs in newio(III) put string
	fread in newio(III) read from file
df(VIII) disk	free
	freopen in newio(III) reopen file
	fscanf in newio(III) input conversion
interactive repair...	fsck(VIII) file system consistency check and
	fseek in newio(III) seek to offset
	fstat(II) get status of open file
	fs(V) format of UNIX file system volume
	ftell in newio(III) get current offset
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
	fwrite in newio(III) write to file
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
dr(IV) DR-11	general device interface
abort(III)	generate an IOT fault
agen(VI)	generate associative memory drivers
glob(VIII)	generate command arguments
ncheck(VIII)	generate names from i-numbers
lex(VI)	generate programs for simple lexical tasks
get(I) get	generation from SCCS file
rand, srand(III) random number	generator
fgetc in newio(III)	get character
getc in newio(III)	get character
getchar in newio(III)	get character
getarg, iargc(III)	get command arguments from Fortran
ftell in newio(III)	get current offset
time(II)	get date and time
nlist(III)	get entries from name list
fileno in newio(III)	get file descriptor
tell(II)	get file offset
stat(II)	get file status
get(I)	get generation from SCCS file
getgid(II)	get group identifications
getpw(III)	get name from UID
getpw in newio(III)	get password line
getpid, getppid(II)	get process identification
times(II)	get process times
fstat(II)	get status of open file
fgets in newio(III)	get string
gets in newio(III)	get string
tty(I)	get terminal name
ino(VIII)	get the i-number of a file
gtty(II)	get typewriter status
getuid(II)	get user identifications
getw in newio(III)	get word
	getarg, iargc(III) get command arguments from Fortran
	getc, getw, fopen(III) buffered input
	getc in newio(III) get character
	getchar in newio(III) get character
	getchar(III) read character
	getgid(II) get group identifications
	get(I) get generation from SCCS file
	getpid, getppid(II) get process identification
getpid,	getppid(II) get process identification
	getpw in newio(III) get password line
	getpw(III) get name from UID
	gets in newio(III) get string
	getty(VIII) set typewriter mode
	getuid(II) get user identifications
getc,	getw, fopen(III) buffered input
	getw in newio(III) get word
	glob(VIII) generate command arguments
ctime, localtime,	gmtime(III) convert date and time to ASCII
reset, setexit(III) execute non-local	goto
greek(VII)	graphics for extended TTY-37 type-box
	greek(VII) graphics for extended TTY-37 type-box
	grep(I) search a file for a pattern
getgid(II) get	group identifications
setgid(II) set process	group ID
chown(II) change owner and	group of a file

newgrp(I) log in to a new	group
gsi(VI) interpret extended character set on	GSI terminal
	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
help(I) ask for	help
	help(I) ask for help
hmul(III)	high-order product
wtmp(V) user login	history
	hmul(III) high-order product
	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
hyphen(VI) find	hyphenated words
	hyphen(VI) find hyphenated words
hypot(III) calculate	hypotenuse
	hypot(III) calculate hypotenuse
getarg,	iargc(III) get command arguments from Fortran
	icheck(VIII) file system storage consistency check
getpid, getppid(II) get process	identification
getgid(II) get group	identifications
getuid(II) get user	identifications
what(I)	identify SCCS files
setgid(II) set process group	ID
setuid(II) set process user	ID
	ierror(III) catch Fortran errors
signal(II) catch or	ignore signals
core(V) format of core	image file
nohup(I) run a command	immune to hangups
	include(V) system data structure definitions file
interface to low speed asynchronous devices	including typewriters...tty(IV)
dump(V)	incremental dump tape format
dump(VIII)	incremental file system dump
restor(VIII)	incremental file system restore
pause(II) suspend execution	indefinitely
ptx(VI) permuted	index
indir(II)	indirect system call
	indir(II) indirect system call
loginfo(II) login	inform.: name, dir, tty, post; udata
utmp(V) user	information
ttys(V) typewriter	initialization data
init(VIII) process control	initialization
	init(VIII) process control initialization
clri(VIII) clear	i-node
	ino(VIII) get the i-number of a file
fscanf in newio(III)	input conversion
scanf in newio(III)	input conversion
sscanf in newio(III)	input conversion
getc, getw, fopen(III) buffered	input
floating point to double precision	integer conversion...dtol(III)
itol(III) integer to long	integer conversion
ltoi(III) long integer to	integer conversion
ltod(III) double precision	integer to floating point conversion
ltoi(III) long	integer to integer conversion
itol(III)	integer to long integer conversion
atoi(III) convert ASCII to	integer
bc(I) arbitrary precision	interactive language
file system consistency check and	interactive repair...fsck(VIII)
dsw(I) delete	interactively
typewriters...tty(IV)	interface to low speed asynchronous devices including
dc(IV) DC-11 communications	interface
dn(IV) DN-11 ACU	interface
dp(IV) DP-11 201 data-phone	interface
dr(IV) DR-11 general device	interface
ht(IV) RH-11/TU-16 magtape	interface

kl(IV) KL-11 or DL-11 asynchronous	interface
tm(IV) TM-11/TU-10 magtape	interface
spline(VI)	interpolate smooth curve
gsi(VI)	interpret extended character set on GSI terminal
fptrap(III) floating point	interpreter
sno(VI) Snobol	interpreter
pipe(II) create an	interprocess channel
return(I) terminate profile or	interrupt processing routine
sleep(I) suspend execution for an	interval
sleep(II) stop execution for	interval
Intro(IV)	INTROD. TO DRIVERS
Intro(V)	INTROD. TO FILE FORMATS
Intro(II)	INTROD. TO MERT FILE FORMATS
Intro(III)	INTROD. TO SUBROUTINES
Intro(VIII)	INTROD. TO SYSTEM PROGRAMS
	intss in newio(III) test for tss or batch
ino(VIII) get the	i-number of a file
ncheck(VIII) generate names from	i-numbers
newio(III) a new	IO subroutine package
abort(III) generate an	IOT fault
	isalpha in newio(III) test for alphabetic
	isdigit in newio(III) test for numeric
	islower in newio(III) test for lower case
	isspace in newio(III) test for space
	isupper in newio(III) test for upper case
continue in sh(I) next	iteration in loop
	itol(III) integer to long integer conversion
bj(VI) the game of black	jack
	join(VI) relational data base operator
search a file for lines containing	keywords...fgrep(VI)
	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
bc(I) arbitrary precision interactive	language
sh(I) shell command programming	language
end, etext, edata(III)	last locations in program
tail(I) deliver the	last part of a file
call,	lcall, vcall(II) create and execute a new process
	ld(I) link editor
strlen in newio(III) obtain string	length
lex(VI) generate programs for simple	lexical tasks
	lex(VI) generate programs for simple lexical tasks
ar(V) archive	(library) file format
ar(I) archive and	library maintainer
read(I) read one	line at a time
col(VI) filter reverse	line feeds
cut(VI) cut out selected fields of each	line of a file
lpd(VIII)	line printer daemon
lpr(I)	line printer spooler
lp(IV)	line printer
getpw in newio(III) get password	line
comm(I) print	lines common to two files
egrep(VI) search a file for	lines containing a pattern
fgrep(VI) search a file for	lines containing keywords
uniq(I) report repeated	lines in a file
rev(VI) reverse	lines of a file
paste(VI) merge the same	lines of all files
a.out(V) assembler and	link editor output
ld(I)	link editor
link(II)	link to a file
dmc(IV) network	link with DDCMP protocol
	link(II) link to a file
ln(I) make a	link
	lint(I) a C program verifier

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
	lnxx(III) return name of current terminal
ctime,	localtime, gmtime(III) convert date and time to ASCII
end, etext, edata(III) last	locations in program
	lock(II) semaphore operations
	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
loginfo(II)	login inform.: name, dir, tty, post; udata
passwd(I) change	login password
	loginfo(II) login inform.: name, dir, tty, post; udata
	login(I) sign onto UNIX
itol(III) integer to	long integer conversion
ltoi(III)	long integer to integer conversion
lseek(III) seek using a	long offset
locv(III)	long output conversion
break in sh(I) exit from	loop
continue in sh(I) next iteration in	loop
nice(I) run a command at	low priority
tty(IV) interface to	low speed asynchronous devices including typewriters
islower in newio(III) test for	lower case
tolower in newio(III) translate to	lower case
	lpd(VIII) line printer daemon
	lp(IV) line printer
	lpr(I) line printer spooler
	lseek(III) seek using a long offset
	ls(I) list contents of directory
conversion...	ltod(III) double precision integer to floating point
	ltoi(III) long integer to integer conversion
	m4(VI) macro processor
wdleng in newio(III) find	machine word size
m4(VI)	macro processor
tmac(VI)	macros for formatting manuscripts
mtm(I)	magnetic tape manipulation
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
make(I)	make a program
mktemp(III)	make a unique named temporary file
delta(I)	make an SCCS delta
cref(I)	make cross reference listing
mkpt(VIII)	make prototype file for use by mkfs
	make(I) make a program
	man(I) print on-line documentation
tp(I)	manipulate DECtape and magtape
mtm(I) magnetic tape	manipulation
man(V)	manual page format
tmac(VI) macros for formatting	manuscripts
	man(V) manual page format
ascii(VII)	map of ASCII character set
neqn(I) typeset	mathematics on terminal
eqn(I) typeset	mathematics

	mem, kmem, null(IV) core memory
agen(VI) generate associative	memory drivers
calloc in newio(III) allocate	memory
cfree in newio(III) deallocate	memory
mem, kmem, null(IV) core	memory
sort(I) sort or	merge files
paste(VI)	merge the same lines of all 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
msg(II) send and receive	messages
sys_nerr, errno(III) system error	messages...perror, sys_errlist,
	mkdir(I) make a directory
mkpt(VIII) make prototype file for use by	mkfs
	mkfs(VIII) construct a file system
	mknod(II) make a directory or a special file
	mknod(VIII) build special file
	mkpt(VIII) make prototype file for use by mkfs
	mktemp(III) make a unique named temporary 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(VII)	mounted file system table
	mount(II) mount file system
	mount(VIII) mount file system
mvall(I)	move all files to a directory
mv(I)	move or rename a file
seek(II)	move read/write pointer
rp(IV) RP-11/RP03	moving-head disk
tmac(VI)	ms macros for formatting manuscripts
	msg(II) send and receive messages
	mtab(VII) mounted file system table
	mtm(I) magnetic tape manipulation
dh(IV) DH-11 communications	multiplexer
	mvall(I) move all files to a directory
	mv(I) move or rename a file
loginfo(II) login inform.:	name, dir, tty, post; udata
getpw(III) get	name from UID
nlist(III) get entries from	name list
nm(I) print	name list
lnxx(III) return	name of current terminal
mktemp(III) make a unique	named temporary file
pwd(I) working directory	name
ncheck(VIII) generate	names from i-numbers
setfil(III) specify Fortran file	name
tmpnam in newio(III) create tmp	name
tty(I) get terminal	name
log(III)	natural logarithm
	ncheck(VIII) generate names from i-numbers
	neqn(I) typeset mathematics on terminal
dmc(IV)	network link with DDCMP protocol
	newgrp(I) log in to a new group
continue in sh(I)	next iteration in loop
	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, troff(I) text formatters
	nroff, troff(I) text formatters
mem, kmem,	null(IV) core memory
rand, srand(III) random	number generator
isdigit in newio(III) test for	numeric
size(I) size of an	object file
reloc(VIII) relocate	object files
strlen in newio(III)	obtain string length
od(I)	octal dump
	od(I) octal dump
fseek in newio(III) seek to	offset
ftell in newio(III) get current	offset
lseek(III) seek using a long	offset
tell(II) get file	offset
read(I) read	one line at a time
man(I) print	on-line documentation
login(I) sign	onto UNIX
dup(II) duplicate an	open file descriptor
fopen in newio(III)	open file
fstat(II) get status of	open file
open(II)	open for reading or writing
	open(II) open for reading or writing
lock(II) semaphore	operations
join(VI) relational data base	operator
stty(I) set terminal	options
rk(IV) RK-11/RK03	(or RK05) disk
cut(VI) cut	out selected fields of each line of a file
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
newio(III) a new IO subroutine	package
man(V) manual	page format
readonly in sh(I) set	parameters to readonly
set in sh(I) set	parameters
tail(I) deliver the last	part of a file
	passwd(I) change login password
	passwd(V) password file
crypt(III)	password encoding
passwd(V)	password file
getpw in newio(III) get	password line
passwd(I) change login	password
	paste(VI) merge the same lines of all files
search a file for lines containing a	pattern...egrep(VI)
grep(I) search a file for a	pattern
	pause(II) suspend execution indefinitely
update(VIII)	periodically update the super block
mesg(I)	permit or deny messages
ptx(VI)	permuted index
error messages...	perror, sys_errlist, sys_nerr, errno(III) system
split(I) split a file into	pieces
tee(I)	pipe fitting
	pipe(II) create an interprocess channel
double precision integer to floating	point conversion...ltod(III)
fptrap(III) floating	point interpreter
dtol(III) floating	point to double precision integer conversion
seek(II) move read/write	pointer
typo(I) find	possible typos
loginfo(II) login inform.: name, dir, tty,	post; udata
	pow(III) floating exponentiation
dtol(III) floating point to double	precision integer conversion
ltod(III) double	precision integer to floating point conversion
bc(I) arbitrary	precision interactive language

monitor(III)	prepare execution profile
	pr(I) print file
date(I)	print and set the date
cal(VI)	print calendar
pr(I)	print file
fprintf in newio(III)	print formatted
printf in newio(III)	print formatted
sprintf in newio(III)	print formatted
comm(I)	print lines common to two files
nm(I)	print name list
man(I)	print on-line documentation
prt(I)	print SCCS file
cat(I) concatenate and	print
lpd(VIII) line	printer daemon
lpr(I) line	printer spooler
lp(IV) line	printer
	printf in newio(III) print formatted
	printf(III) formatted 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) MERT startup
abort in newio(III) abort	process
lcall, vcall(II) create and execute a new	process...call,
return(I) terminate profile or interrupt	processing routine
m4(VI) macro	processor
hmul(III) high-order	product
	prof(I) display profile data
prof(I) display	profile data
return(I) terminate	profile or interrupt processing routine
monitor(III) prepare execution	profile
profil(II) execution time	profile
	profil(II) execution time profile
Intro(VIII) INTROD. TO SYSTEM	PROGRAMS
dmc(IV) network link with DDCMP	protocol
mkpt(VIII) make	prototype file for use by mkfs
	prt(I) print SCCS file
	ps(I) process status
	ptx(VI) permuted index
ungetc in newio(III)	push character back
fputc in newio(III)	put character
putc in newio(III)	put character
putchar in newio(III)	put character
fputs in newio(III)	put string
puts in newio(III)	put string
putw in newio(III)	put word
	putc in newio(III) put character
	putc, putw, fcreat, fflush(III) buffered output
	putchar, flush(III) write character
	putchar in newio(III) put character
	puts in newio(III) put string
putc,	putw, fcreat, fflush(III) buffered output
	putw in newio(III) put word
	pwd(I) working directory name
compar(III) default comparison routine for	qsort
	qsort(III) quicker sort
qsort(III)	quicker sort
	rand, srand(III) random number generator
rand, srand(III)	random number generator
rc(VI)	Ratfor compiler
	rc(VI) Ratfor compiler
getchar(III)	read character
csw(II)	read console switches
fread in newio(III)	read from file
read(II)	read from file

read(I)	read one line at a time
	read(I) read one line at a time
	read(II) read from file
open(II) open for	reading or writing
	readonly in sh(I) set parameters to readonly
readonly in sh(I) set parameters to	readonly
seek(II) move	read/write pointer
msg(II) send and	receive messages
cref(I) make cross	reference listing
reform(VI)	reformat text file
	reform(VI) reformat text file
join(VI)	relational data base operator
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
deroff(VI)	remove Troff and Eqn constructs
rm(I)	remove (unlink) files
mv(I) move or	rename a file
freopen in newio(III)	reopen file
system consistency check and interactive	repair...fsck(VIII) 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
lnxx(III)	return name of current terminal
routine...	return(I) terminate profile or interrupt processing
col(VI) filter	reverse line feeds
rev(VI)	reverse lines of a file
	rev(VI) reverse lines of a file
	rew(I) rewind tape
	rewind in newio(III) rewind
rew(I)	rewind tape
rewind in newio(III)	rewind
rf(IV)	RF11/RS11 fixed-head disk file
	rf(IV) RF11/RS11 fixed-head disk file
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
chroot(I) change	root directory for a command
sqrt(III) square	root function
compar(III) default comparison	routine for qsort
terminate profile or interrupt processing	routine...return(I)
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
paste(VI) merge the	same lines of all files
sdh(IV) DH11 for	Satellite Processor System
break, brk,	sbrk(II) change core allocation
	scanf in newio(III) input conversion
delta(I) make an	SCCS delta
get(I) get generation from	SCCS file
prt(I) print	SCCS file
admin(I) administer	SCCS files
sccsfile(V) format of	SCCS file
what(I) identify	SCCS files
	sccsfile(V) format of SCCS file
	sdh(IV) DH11 for Satellite Processor System

grep(I)	search a file for a pattern
egrep(VI)	search a file for lines containing a pattern
fgrep(VI)	search a file for lines containing keywords
	sed(I) stream editor
fseek in newio(III)	seek to offset
lseek(III)	seek using a long offset
	seek(II) move read/write pointer
cut(VI) cut out	selected fields of each line of a file
lock(II)	semaphore operations
msg(II)	send and receive messages
mail(I)	send mail to designated users
kill(II)	send signal to a process
setbuf in newio(III)	set buffer size
	set in sh(I) set parameters
stty(II)	set mode of typewriter
gsi(VI) interpret extended character	set on GSI terminal
readonly in sh(I)	set parameters to readonly
set in sh(I)	set parameters
setgid(II)	set process group ID
setuid(II)	set process user ID
nice(II)	set program priority
tabs(VII)	set tab stops
tabs(VI)	set tabs on terminal
stty(I)	set terminal options
date(I) print and	set the date
stime(II)	set time
getty(VIII)	set typewriter mode
ascii(VII) map of ASCII character	set
	setbuf in newio(III) set buffer size
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
shift(I) adjust	Shell arguments
sh(I)	shell command programming language
exec in sh(I) execute within	shell
	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
trap in sh(I) catch	signals
dirname(I) strip	simple filename
lex(VI) generate programs for	simple lexical tasks
	sin, cos(III) trigonometric functions
size(I)	size of an object file
	size(I) size of an object file
setbuf in newio(III) set buffer	size
wdleng in newio(III) find machine word	size
	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
sort(I)	sort or merge files
	sort(I) sort or merge files
qsort(III) quicker	sort
isspace in newio(III) test for	space
fork(II)	spawn new process
mknod(II) make a directory or a	special file
mknod(VIII) build	special file
setfil(III)	specify Fortran file name
tty(IV) interface to low	speed asynchronous devices including typewriters
spell(VI) find	spelling errors
	spell(VI) find spelling errors
	spline(VI) interpolate smooth curve

split(I)	split a file into pieces
	split(I) split a file into pieces
lpr(I) line printer	spooler
	sprintf in newio(III) print formatted
	sqrt(III) square root function
sqrt(III)	square root function
rand,	srand(III) random number generator
	sscanf in newio(III) input conversion
boot procedures(VIII) MERT	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
icheck(VIII) file system	storage consistency check
	strcat in newio(III) concatenate strings
	strcmp in newio(III) compare strings
	strcpy in newio(III) copy string
sed(I)	stream editor
strlen in newio(III) obtain	string length
fgets in newio(III) get	string
fputs in newio(III) put	string
gets in newio(III) get	string
puts in newio(III) put	string
strcat in newio(III) concatenate	strings
strcmp in newio(III) compare	strings
strcpy in newio(III) copy	string
basename(I)	strip filename affixes
dirname(I)	strip simple filename
	strip(I) remove symbols and relocation bits
	strlen in newio(III) obtain string length
include(V) system data	structure definitions file
	stty(I) set terminal options
	stty(II) set mode of typewriter
newio(III) a new IO	subroutine package
exit in newio(III) exit from	subroutine
Intro(III) INTROD. TO	SUBROUTINES
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
pause(II)	suspend execution indefinitely
	su(VIII) become privileged user
csw(II) read console	switches
strip(I) remove	symbols and relocation bits
du(IV) DU-11	synchronous communication device
	sync(II) update super-block
	sync(VIII) update the super block
messages...perror,	sys_errlist, sys_nerr, errno(III) system error
perror, sys_errlist,	sys_nerr, errno(III) system error messages
indir(II) indirect	system call
fsck(VIII) file	system consistency check and interactive repair
check(VIII) file	system consistency check
crash(VIII) what to do when the	system crashes
include(V)	system data structure definitions file
dcheck(VIII) file	system directory consistency check
dump(VIII) incremental file	system dump
sys_errlist, sys_nerr, errno(III)	system error messages...perror,
	system in newio(III) execute command
Intro(VIII) INTROD. TO	SYSTEM PROGRAMS

restor(VIII) incremental file	system restore
icheck(VIII) file	system storage consistency check
mtab(VII) mounted file	system table
fs(V) format of UNIX file	system volume
sdh(IV) DH11 for Satellite Processor	System
tabs(VII) set	tab stops
mtab(VII) mounted file system	table
tbl(VI) format	tables for nroff or troff
tabs(VI) set	tabs on terminal
	tabs(VI) set tabs on terminal
	tabs(VII) set tab stops
	tail(I) deliver the last part of a file
atan, atan2(III) arc	tangent function
dump(V) incremental dump	tape format
tp(V) DEC/mag	tape formats
mtm(I) magnetic	tape manipulation
rew(I) rewind	tape
generate programs for simple lexical	tasks...lex(VI)
	tbl(VI) format tables for nroff or troff
tc(IV)	TC-11/TU56 DECtape
	tc(IV) TC-11/TU56 DECtape
	tee(I) pipe fitting
tf(IV)	Telefile disk driver
	tell(II) get file offset
mktemp(III) make a unique named	temporary file
tty(I) get	terminal name
stty(I) set	terminal options
interpret extended character set on GSI	terminal...gsi(VI)
lnxx(III) return name of current	terminal
neqn(I) typeset mathematics on	terminal
tabs(VI) set tabs on	terminal
kill(I)	terminate a process
exit(I)	terminate command file
exit(II)	terminate process
return(I)	terminate profile or interrupt processing routine
wait(II) wait for process to	terminate
wait in sh(I) wait for process	termination
isalpha in newio(III)	test for alphabetic
islower in newio(III)	test for lower case
isdigit in newio(III)	test for numeric
isspace in newio(III)	test for space
intss in newio(III)	test for tss or batch
isupper in newio(III)	test for upper case
	test(I) condition command
ed(I)	text editor
reform(VI) reformat	text file
nroff, troff(I)	text formatters
nroff, troff(I)	text formatters
	tf(IV) Telefile disk driver
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
alarm(II) activate alarm clock	timer
read(I) read one line at a	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
	tmac(VI) ms macros for formatting manuscripts
	tm(IV) TM-11/TU-10 magtape interface

tmpnam in newio(III) create	tmp name
	tmpnam in newio(III) create tmp name
	tolower in newio(III) translate to lower case
	toupper in newio(III) translate to upper case
	tp(I) manipulate DECtape and magtape
	tp(V) DEC/mag tape formats
tolower in newio(III)	translate to lower case
toupper in newio(III)	translate to upper case
tr(I)	transliterate
	trap in sh(I) catch signals
	tr(I) transliterate
sin, cos(III)	trigonometric functions
deroff(VI) remove	Troff and Eqn constructs
nroff,	troff(I) text formatters
nroff,	troff(I) text formatters
tbl(VI) format tables for nroff or	troff
intss in newio(III) test for	tss or batch
	ttt(VI) the game of tic-tac-toe
loginfo(II) login inform.: name, dir,	tty, post; udata
greek(VII) graphics for extended	TTY-37 type-box
	tty(I) get terminal name
including typewriters...	tty(IV) interface to low speed asynchronous devices
	ttys(V) typewriter initialization data
cmp(I) compare	two files
comm(I) print lines common to	two files
greek(VII) graphics for extended TTY-37	type-box
neqn(I)	typeset mathematics on terminal
eqn(I)	typeset mathematics
ttys(V)	typewriter initialization data
getty(VIII) set	typewriter mode
gtty(II) get	typewriter status
mesg(III) write message on	typewriter
stty(II) set mode of	typewriter
to low speed asynchronous devices including	typewriters...tty(IV) interface
	typo(I) find possible typos
typo(I) find possible	typos
login inform.: name, dir, tty, post;	udata...loginfo(II)
getpw(III) get name from	UID
	umount(II) dismount file system
	umount(VIII) dismount file system
	ungetc in newio(III) push character back
	uniq(I) report repeated lines in a file
mktemp(III) make a	unique named temporary file
	units(VI) conversion program
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
isupper in newio(III) test for	upper case
toupper in newio(III) translate to	upper case
du(I) summarize disk	usage
mkpt(VIII) make prototype file for	use by mkfs
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(I) write to all	users
wall(VIII) write to all	users
write(I) write to another	user
lseek(III) seek	using a long offset
	utmp(V) user information
	uucp(VI) unix-to-unix copy

abs, fabs(III) absolute	value
call, lcall,	vcall(II) create and execute a new process
lint(I) a C program	verifier
fs(V) format of UNIX file system	volume
wait in sh(I)	wait for process termination
wait(II)	wait for process to terminate
	wait in sh(I) wait for process termination
	wait(I) await completion of process
	wait(II) wait for process to terminate
	wall(I) write to all users
	wall(VIII) write to all users
	wc(I) word count
	wdleng in newio(III) find machine word size
crash(VIII)	what to do when the system crashes
	what(I) identify SCCS files
who(I)	who is on the system
	who(I) who is on the system
exec in sh(I) execute	within shell
wc(I)	word count
wdleng in newio(III) find machine	word size
getw in newio(III) get	word
putw in newio(III) put	word
hyphen(VI) find hyphenated	words
pwd(I)	working directory name
chdir, cd(I) change	working directory
chdir(II) change	working directory
putchar, flush(III)	write character
mesg(III)	write message on typewriter
write(II)	write on a file
wall(I)	write to all users
wall(VIII)	write to all users
write(I)	write to another user
fwrite in newio(III)	write to file
	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