UNIX User's Reference Manual
(URM)

EUNICE BSD
Version 4.3.2

August 1988

The Wollongong Group, Inc.
1129 San Antonio Road
Palo Alto, California 94303
(415) 962-7100
TWX 910-373-2085

Restricted Rights

Copyrights

This manual reflects system enhancements made at Berkeley and sponsored in part by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089. The views and conclusions contained in these documents are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency or of the US Government.

Trademarks

PREFACE

This update to the 4.2 distribution of August 1983 provides substantially improved performance, reliability, and security, the addition of Xerox Network System (NS) to the set of networking domains, and partial support for the VAX 8600 and MICROVAXII.

We were greatly assisted by the DEC UNIX Engineering group who provided two full time employees, Miriam Amos and Kevin Dunlap, to work at Berkeley. They were responsible for developing and debugging the distributed domain based name server and integrating it into the mail system. Mt Xinu provided the bug list distribution service as well as donating their MICROVAXII port to 4.3BSD. Drivers for the MICROVAXII were done by Rick Macklem at the University of Guelph. Sam Leffler provided valuable assistance and advice with many projects. Keith Sklower coordinated with William Nesheim and J. Q. Johnson at Cornell, and Chris Torek and James O'Toole at the University of Maryland to do the Xerox Network Systems implementation. Robert Elz at the University of Melbourne contributed greatly to the performance work in the kernel. Donn Seeley and Jay Lepreau at the University of Utah relentlessly dealt with a miriad of details; Donn completed the unfinished performance work on Fortran 77 and fixed numerous C compiler bugs. Ralph Campbell handled innumerable questions and problem reports and had time left to write rdist. George Goble was invaluable in shaking out the bugs on his production systems long before we were confident enough to inflict it on our users. Bill Shannon at Sun Microsystems has been helpful in providing us with bug fixes and improvements. Tom Ferrin, in his capacity as Board Member of Usenix Association, handled the logistics of large-scale reproduction of the 4.2BSD and 4.3BSD manuals. Mark Seiden helped with the typesetting and indexing of the 4.3BSD manuals. Special mention goes to Bob Henry for keeping ucbvax running in spite of new and improved software and an ever increasing mail, news, and uucp load.

Numerous others contributed their time and energy in creating the user contributed software for the release. As always, we are grateful to the UNIX user community for encouragement and support.

Once again, the financial support of the Defense Advanced Research Projects Agency is gratefully acknowledged.

M. K. McKusick
M. J. Karels
J. M. Bloom

Preface to the 4.2 Berkeley distribution

This update to the 4.1 distribution of June 1981 provides support for the VAX 11/730, full networking and interprocess communication support, an entirely new file system, and many other new features. It is certainly the most ambitious release of software ever prepared here and represents many man-years of work. Bill Shannon (both at DEC and at Sun Microsystems) and Robert Elz of the University of Melbourne contributed greatly to this distribution through new device drivers and painful debugging episodes. Rob Gurwitz of BBN wrote the initial version of the code upon which the current networking support is based. Eric Allman of Britton-Lee donated countless hours to the mail system. Bill Croft (both at SRI and Sun Microsystems) aided in the debugging and development of the networking facilities. Dennis Ritchie of Bell Laboratories also contributed greatly to this distribution, providing valuable advise and guidance. Helge Skrivervik worked on the device drivers which enabled the distribution to be delivered with a TU58 console cassette and RX01 console floppppy disk, and rewrote major portions of the standalone i/o system to support formatting of non-DEC peripherals.

Numerous others contributed their time and energy in organizing the user software for release, while many groups of people on campus suffered patiently through the low spots of development. As always, we are grateful to the UNIX user community for encouragement and support.

Once again, the financial support of the Defense Advanced Research Projects Agency is gratefully acknowledged.

S. J. Leffler
W. N. Joy
M. K. McKusick

Preface to the 4.1 Berkeley distribution

This update to the fourth distribution of November 1980 provides support for the VAX 11/750 and for the full interconnect architecture of the VAX 11/780. Robert Elz of the University of Melbourne contributed greatly to this distribution especially in the boot-time system configuration code; Bill Shannon of DEC supplied us with the implementation of DEC standard bad block handling. The research group at Bell Laboratories and DEC Merrimack provided us with access to 11/750's in order to debug its support.

Other individuals too numerous to mention provided us with bug reports, fixes and other enhancements which are reflected in the system. We are grateful to the UNIX user community for encouragement and support.

The financial support of the Defence Advanced Research Projects Agency in support of this work is gratefully acknowledged.

W. N. Joy
R. S. Fabry
K. Sklower

Preface to the Fourth Berkeley distribution

This manual reflects the Berkeley system mid-October, 1980. A large amount of tuning has been done in the system since the last release; we hope this provides as noticeable an improvement for you as it did for us. This release finds the system in transition; a number of facilities have been added in experimental versions (job control, resource limits) and the implementation of others is imminent (shared-segments, higher performance from the file system, etc.). Applications which use facilities that are in transition should be aware that some of the system calls and library routines will change in the near future. We have tried to be conscientious and make it very clear where this is likely.

A new group has been formed at Berkeley, to assume responsibility for the future development and support of a version of UNIX on the VAX. The group has received funding from the Defense Advanced Research Projects Agency (DARPA) to supply a standard version of the system to DARPA contractors. The same version of the system will be made available to other licensees of UNIX on the VAX for a duplication charge. We gratefully acknowledge the support of this contract.

We wish to acknowledge the contribution of a number of individuals to the the system.

We would especially like to thank Jim Kulp of IIASA, Laxenburg Austria and his colleagues, who first put job control facilities into UNIX; Eric Allman, Robert Henry, Peter Kessler and Kirk McKusick, who contributed major new pieces of software; Mark Horton, who contributed to the improvement of facilities and substantially improved the quality of our bit-mapped fonts, our hardware support staff: Bob Kridle, Anita Hirsch, Len Edmondson and Fred Archibald, who helped us to debug a number of new peripherals; Ken Arnold who did much of the leg-work in getting this version of the manual prepared, and did the final editing of sections 2-6, some special individuals within Bell Laboratories: Greg Chesson, Stuart Feldman, Dick Haight, Howard Katseff, Brian Kernighan, Tom London, John Reiser, Dennis Ritchie, Ken Thompson, and Peter Weinberger who helped out by answering questions; our excellent local DEC field service people, Kevin Althaus and Frank Chargois who kept our machine running virtually all the time, and fixed it quickly when things broke; and, Mike Accetta of Carnegie-Mellon University, Robert Elz of the University of Melbourne, George Goble of Purdue University, and David Kashtan of the Stanford Research Institute for their technical advice and support.

Special thanks to Bill Munson of DEC who helped by augmenting our computing facility and to Eric Allman for carefully proofreading the "last" draft of the manual and finding the bugs which we knew were there but couldn't see.

We dedicate this to the memory of David Sakrison, late chairman of our department, who gave his support to the establishment of our VAX computing facility, and to our department as a whole.

W. N. Joy
Ö. Babaoğlu
R. S. Fabry
K. Sklower

Preface to the Third Berkeley distribution

This manual reflects the state of the Berkeley system, December 1979. We would like to thank all the people at Berkeley who have contributed to the system, and particularly thank Prof. Richard Fateman for creating and administrating a hospitable environment, Mark Horton who helped prepare this manual, and Eric Allman, Bob Kridle, Juan Porcar and Richard Tuck for their contributions to the kernel.

The cooperation of Bell Laboratories in providing us with an early version of UNIX/32V is greatly appreciated. We would especially like to thank Dr. Charles Roberts of Bell Laboratories for helping us obtain this release, and acknowledge T. B. London, J. F. Reiser, K. Thompson, D. M. Ritchie, G. Chesson and H. P. Katseff for their advice and support.

W. N. Joy
Ö. Babaoğlu

Preface to the UNIX/32V distribution

The UNIX operating system for the VAX-11 provides substantially the same facilities as the UNIX system for the PDP*-11.

We acknowledge the work of many who came before us, and particularly thank G. K. Swanson, W. M. Cardoza, D. K. Sharma, and J. F. Jarvis for assistance with the implementation for the VAX-11/780.

T. B. London
J. F. Reiser

Preface to the Seventh Edition

Although this Seventh Edition no longer bears their byline, Ken Thompson and Dennis Ritchie remain the fathers and preceptors of the UNIX time-sharing system. Many of the improvements here described bear their mark. Among many, many other people who have contributed to the further flowering of UNIX, we wish especially to acknowledge the contributions of A. V. Aho, S. R. Bourne, L. L. Cherry, G. L. Chesson, S. I. Feldman, C. B. Haley, R. C. Haight, S. C. Johnson, M. E. Lesk, T. L. Lyon, L. E. McMahon, R. Morris, R. Muha, D. A. Nowitz, L. Wehr, and P. J. Weinberger. We appreciate also the effective advice and criticism of T. A. Dolotta, A. G. Fraser, J. F. Maranzano, and J. R. Mashey; and we remember the important work of the late Joseph F. Ossanna.

B. W. Kernighan
M. D. McIlroy

INTRODUCTION TO USER'S REFERENCE MANUAL

The documentation has been reorganized for 4.3 BSD in a format similar to the one used for the Usenix 4.2BSD manuals. It is divided into three sets; each set consists of one or more volumes. The abbreviations for the volume names are listed in square brackets; the abbreviations for the manual sections are listed in parenthesis.

- I. User's Documents
  - User's Reference Manual [URM]
    - Commands (1)
    - Games (6)
    - Macro packages and language conventions (7)
  - User's Supplementary Documents [USD]
    - Getting Started
    - Basic Utilities
    - Communicating with the World
    - Text Editing
    - Document Preparation
    - Amusements
- II. Programmer's Documents
  - Programmer's Reference Manual [PRM]
    - System calls (2)
    - Subroutines (3)
    - Special files (4)
    - File formats and conventions (5)
  - Programmer's Supplementary Documents, Volume 1 [PS1]
    - Languages in common use
    - General Reference
    - Programming Tools
    - Programming Libraries
  - Programmer's Supplementary Documents, Volume 2 [PS2]
    - Documents of Historic Interest
    - Other Languages
    - Database Management
- III. System Manager's Manual [SMM]
  - Maintenance commands (8)
  - System Installation and Administration
  - Supporting Documentation

References to individual documents are given as "volume:document", thus USD:1 refers to the first document in the "User's Supplementary Documents". References to manual pages are given as "*name*(section)" thus *sh*(1) refers to the shell manual entry in section 1.

The manual pages give descriptions of the publicly available features of the UNIX/32V system, as extended to provide a virtual memory environment and other enhancements at the University of California. They do not attempt to provide perspective or tutorial information about the UNIX operating system, its facilities, or its implementation. Various documents on those topics are contained in the "UNIX User's Supplementary Documents" (USD), the "UNIX Programmer's Supplementary

Documents'' (PS1 and PS2), and ''UNIX System Manager's Manual'' (SMM). In particular, for an overview see ''The UNIX Time-Sharing System'' (PS2:1) by Ritchie and Thompson; for a tutorial see ''UNIX for Beginners'' (USD:1) by Kernighan, and for a guide to the new features of this virtual version, see ''Berkeley Software Architecture Manual (4.3 Edition)'' (PS1:6).

Within the area it surveys, this volume attempts to be timely, complete and concise. Where the latter two objectives conflict, the obvious is often left unsaid in favor of brevity. It is intended that each program be described as it is, not as it should be. Inevitably, this means that various sections will soon be out of date.

Commands are programs intended to be invoked directly by the user, in contrast to subroutines, that are intended to be called by the user's programs. User commands are described in URM section 1. Commands generally reside in directory */bin* (for *bin* ary programs). Some programs also reside in */usr/bin*, */usr/ucb*, or */usr/new*, to save space in */bin*. These directories are searched automatically by the command interpreters.

Games have been relegated to URM section 6 and */usr/games*, to keep them from contaminating the more staid information of URM section 1.

Miscellaneous collection of information necessary for writing in various specialized languages such as character codes, macro packages for typesetting, etc is contained in URM section 7.

System calls are entries into the UNIX® supervisor. The system call interface is identical to a C language procedure call; the equivalent C procedures are described in PRM section 2.

An assortment of subroutines is available; they are described in PRM section 3. The primary libraries in which they are kept are described in *intro*(3). The functions are described in terms of C; those that will work with Fortran are described in *intro*(3f).

PRM section 4 discusses the characteristics of each system ''file'' that refers to an I/O device. The names in this section refer to the DEC device names for the hardware, instead of the names of the special files themselves.

The file formats and conventions (PRM section 5) 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.

Commands and procedures intended for use primarily by the system administrator are described in SMM section 8. The commands and files described here are almost all kept in the directory */etc*. EUNICE administration commands are kept in the directory */etc/eunice.*

Each section consists of independent entries of a page or so each. The name of the entry is in the upper corners of its pages, together with the section number, and sometimes a letter characteristic of a subcategory, e.g. graphics is 1G, and the math library is 3M. Entries within each section are alphabetized. except for PRM section 3f which appears after the rest of PRM section 3. The page numbers of each entry start at 1; it is infeasible to number consecutively the pages of a document like this that is republished in many variant forms.

All entries are based on a common format; not all subsections always appear.

> The *name* subsection lists the exact names of the commands and subroutines covered under the entry and gives a short description of their purpose.
>
> The *synopsis* summarizes the use of the program being described. A few conventions are used, particularly in the Commands subsection:
>
>> **Boldface** words are considered literals, and are typed just as they appear.
>>
>> Square brackets [ ] around an argument show 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 ‘‘–’’ usually means that it is an option-specifying 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* subsection discusses in detail the subject at hand.

A section called *eunice notes* has been added to indicate any differences which might exist between the native UNIX implementation of a command and that of a command provided with EUNICE.

The *files* subsection gives the names of files that are built into the program.

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

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

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

At the beginning of URM 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 that exist only to exercise a particular system call.

HOW TO GET STARTED

This section sketches 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’’ in (USD:1) for a more complete introduction to the system.

Please note that the following information is true for native UNIX. Users of EUNICE should refer to the *EUNICE Reference Manual* for further usage information.

Logging in. Almost any ASCII terminal capable of full duplex operation and generating the entire character set can be used. You must have a valid user name, which may be obtained from the system administration. If you will be accessing UNIX remotely, you will also need to obtain the telephone number for the system that you will be using.

After a data connection is established, the login procedure depends on what type of terminal you are using and local system conventions. If your terminal is directly connected to the computer, it generally runs at 9600 or 19200 baud. If you are using a modem running over a phone line, the terminal must be set at the speed appropriate for the modem you are using, typically 300, 1200, or 2400 baud. The half/full duplex switch should always 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 suppresses echo to 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. A message-of-the-day usually greets you before your first prompt.

If the system types out a few garbage characters after you have established a data connection (the ‘‘login:’’ message at the wrong speed), depress the ‘‘break’’ (or ‘‘interrupt’’) key. This is a speed-independent signal to UNIX that a different speed terminal is in use. The system then will type ‘‘login:,’’ this time at another speed. Continue depressing the break key until ‘‘login:’’ appears clearly, then respond with your user name.

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 lower-case letters to upper case.

The evidence that you have successfully logged in is that a shell program will type a prompt (‘‘$’’ or ‘‘%’’) to you. (The shells are described below under ‘‘How to run a program.’’)

For more information, consult *tset*(1), and *stty*(1), which tell how to adjust terminal behavior; *getty*(8) discusses the login sequence in more detail, and *tty*(4) discusses terminal I/O.

Logging out. There are three ways to log out:

> By typing "logout" or 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 log in directly as another user by giving a *login*(1) command.
>
> If worse comes to worse, you can simply hang up the phone; but beware – some machines may lack the necessary hardware to detect that the phone has been hung up. Ask your system administrator if this is a problem on your machine.

How to communicate through your terminal. When you type characters, a gnome deep in the system gathers your characters and saves them in a secret place. The characters will not be given to a program until you type a return (or newline), as described above in *Logging in.*

UNIX terminal 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 printed 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 (or beeps, if your prompt was a "%").

The delete (DEL) character in typed input kills all the preceding characters in the line, so typing mistakes can be repaired on a single line. Also, the backspace character (control-H) erases the last character typed. *Tset*(1) or *stty*(1) can be used to change these defaults. Successive uses of backspace erases characters back to, but not beyond, the beginning of the line. DEL and backspace can be transmitted to a program by preceding them with "\". (So, to erase "\", you need two backspaces).

An *interrupt signal* is sent to a program by typing control-C or the "break" key which is not passed to programs. This signal generally causes whatever program you are running to terminate. It is typically used to stop a long printout that you do not want. 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 interrupt character can also be changed with *tset*(1) or *stty*(1).

It is also possible to suspend output temporarily using ^S (control-S) and later resume output with ^Q (control-Q). Output can be thrown away without interrupting the program by typing ^O (control-O); see *tty*(4).

The *quit* signal is generated by typing the ASCII FS character. (FS appears many places on different terminals, most commonly as control-\ or control-|.) 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 newline function or whether it must be simulated with carriage-return and line-feed. In the latter case, all input carriage returns are turned to newline 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 *reset*(1) command will rescue you. If the terminal does not appear to be echoing anything that you type, it may be stuck in "no-echo" or "raw" mode. Try typing "(control-J)reset(control-J)" to recover.

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 *tset*(1) or *stty*(1) command can be used to change these defaults. *Tset*(1) can be used to set the tab stops automatically when necessary.

How to run a program; the shells. When you have successfully logged in, a program called a 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 in several system directories to find the command. You can also place commands in your own directory and have the shell find them there. 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 prompt at you to show that it is ready for another command.

The shells have many other capabilities, that are described in detail in sections *sh*(1) and *csh*(1). If the shell prompts you with "$", then it is an instance of *sh*(1) the standard shell provided by Bell Labs. If it prompts with "%" then it is an instance of *csh*(1), a shell written at Berkeley. The shells are different for all but the most simple terminal usage. Most users at Berkeley choose *csh*(1) because of the *history* mechanism and the *alias* feature, that greatly enhance its power when used interactively. *Csh* also supports the job-control facilities; see *csh*(1) or the Csh introduction in USD:4 for details.

You can change from one shell to the other by using the *chsh*(1) command, which takes effect at your next login.

The current directory. UNIX has a file system arranged as a hierarchy of directories. When the system administrator gave you a user name, they 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 permission 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 perusal by other users.

To change the current directory (but not the set of permissions you were endowed with at login) use *cd*(1).

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. For example, */usr/tmp/filex* refers to the file *filex* in the directory *tmp; tmp* 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 with no prefixed "/".

A path name may be used anywhere a file name is required.

Important commands that modify the contents of files are *cp*(1), *mv*(1), and *rm*(1), which respectively copy, move (i.e. rename) and remove files. To find out the status of files or directories, use *ls*(1). See *mkdir*(1) for making directories and *rmdir*(1) for destroying them.

For a fuller discussion of the file system, see "A Fast File System for UNIX" (SMM:14) by McKusick, Joy, Leffler, and Fabry. It may also be useful to glance through PRM section 2, that discusses system calls, even if you do not 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 the editor *ex*(1) or its display editing alias *vi*(1). (The old standard editor *ed*(1) is also available.) The principal languages in UNIX are provided by the C compiler *cc*(1), the Fortran compiler *f77*(1), and its derivatives *efl*(1) and *ratfor*(1), the Pascal compiler *pc*(1), and interpreter *pi*(1), and the Lisp system *lisp*(1). User contributed software in the latest release of the system supports APL, B, the Functional Programming language, and Icon. Refer to *apl*(1), *b*(1), *fp*(1), and *icon*(1), respectively for more information about each. After the program text has been entered through the editor and written to 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*(1) to move it to a less exposed name after successful compilation.

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 shell ("$" or "%") prompt.

Your programs can receive arguments from the command line just as system programs do, see "UNIX Programming - Second Edition" (PS2:3), or for a more terse description *execve*(2).

Text processing. Almost all text is entered through the editor *ex*(1) (often entered via *vi*(1)). The commands most often used to write text on a terminal are: *cat*(1), *more*(1), and *nroff*(1).

The *cat*(1) command simply dumps ASCII text on the terminal, with no processing at all. *More*(1) is useful for preventing the output of a command from scrolling off the top of your screen. It is also well suited to perusing files. *Nroff*(1) is an elaborate text formatting program. Used naked, it requires careful forethought, but for ordinary documents it has been tamed; see *me*(7) and *ms*(7).

Troff(1) prepares documents for a Graphics Systems phototypesetter or a Versatec Plotter; it is similar to *nroff*(1), and often works from exactly the same source text. It was used to produce this manual.

Script(1) lets you keep a record of your session in a file, which can then be printed, mailed, etc. It provides the advantages of a hard-copy terminal even when using a display terminal.

Status inquiries. Various commands exist to provide you with useful information. *w*(1) prints a list of users currently logged in, and what they are doing. *date*(1) prints the current time and date. *ls*(1) will list the files in your directory or give summary information about particular files.

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*(1) or *talk*(1) is used; *mail*(1) will leave a message whose presence will be announced to another user when they next log in. The write-ups in the manual also suggest how to respond to the these commands if you are a target.

If you use *csh*(1) the key ^Z (control-Z) will cause jobs to "stop". If this happens before you learn about it, you can simply continue by saying "fg" (for foreground) to bring the job back.

User Contributed Software

On the standard 4.3 UNIX BSD release, the subtree /usr/src/new contains programs contributed by the user community. None of these utilities have been included in the current release of EUNICE. This list is provided so that the users may have a more complete list of commands excluded from EUNICE Version 4.3.2.

Directory	Description	Contributor(s)
B	B programming language & environment	CWI
W	X Window system	M.I.T.
ansi	ANSI and VMS standard tape handler	Tom Quarles, Berkeley
apl	APL system	Purdue
bib	bibliography system	Arizona
courier	remote procedure call package	Eric Cooper, Berkeley
cpm	CP/M floppy access package	Helge Skrivervik
dipress	Xerox Interpress Tools	Xerox
dsh	distributed shell	Dave Presotto, Berkeley
emacs	Gnumacs	Richard Stallman
enet	Packet filter	Jeff Mogul, Stanford
help	Help system	John Kunze, Berkeley
hyper	Hyperchannel support tools	Steve Glaser, Tektronix
icon	ICON system	Arizona
jove	Emacs editor	Jon Payne
kermit	File transfer protocol	Columbia University
mh	MH mail system	Rand Corporation
mkmf	Makefile generator	Peter Nicklin, Berkeley
mmdf	MMDF mail system	Dr. Dave Farber, Delaware
news	"readnews" bulletin board system	Matt Glickman, Berkeley
notes	notes files bulletin board system	Illinois
np100	Utilities for Interlan NP100	MICOM-Interlan
patch	apply diffs to originals	Larry Wall, SDC
pathalias	uucp router	Peter Honeyman, Princeton
rn	readnews front end	Larry Wall
spms	software project management system	Peter Nicklin, Berkeley
sumacc	MacIntosh cross development system	William Croft, Stanford
sunrpc	Remote procedure call package	Sun Microsystems
tac	reverse a file by segments	Jay Lepreau, Utah
tools	miscellaneous tools	John Kunze, Berkeley
umodem	File transfer protocol	Lauren Weinstein
xns	XNS/Courier user code	J.Q. Johnson, Cornell

TABLE OF CONTENTS

1. Commands and Application Programs

2. System Calls

3. C Library Subroutines

3f. Fortran Library

4. Special Files

5. File Formats

6. Games

7. Miscellaneous

8. System Maintenance