

**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

Use, duplication or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in FAR 52.227-7013. The Wollongong Group, Inc., 1129 San Antonio Road, Palo Alto, California 94303, U.S.A.

Copyrights

Copyright © 1982, 1983, 1984, 1985, 1987, 1988 The Wollongong Group, Inc. All rights reserved. This software and its related documents contain confidential trade secret information of The Wollongong Group, Inc. No part of this program or publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, except as provided in the license agreement governing the computer software and documentation or by prior written permission of The Wollongong Group, Inc., 1129 San Antonio Road, Palo Alto, California 94303, U.S.A.

Copyright 1979, 1980, 1983, 1986 Regents of the University of California. Permission to copy these documents or any portion thereof as necessary for licensed use of the software is granted to licensees of this software, provided this copyright notice and statement of permission are included.

Copyright 1979, AT&T Bell Laboratories, Incorporated. Holders of UNIXTM/32V, System III, or System V software licenses are permitted to copy these documents, or any portion of them, as necessary for licensed use of the software, provided this copyright notice and statement of permission are included.

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

EUNICE and WIN are trademarks of The Wollongong Group, Inc.

DEC, PDP, VAX, VMS, and MicroVAX are trademarks of Digital Equipment Corporation.

UNIX is a registered trademark of AT&T.

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 myriad 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 advice and guidance. Helge Skrivervik worked on the device drivers which enabled the distribution to be delivered with a TU58 console cassette and RX01 console floppy 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 ILASA, 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 administering 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
O. 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-l.) 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 *cs(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 *cs(1)*, a shell written at Berkeley. The shells are different for all but the most simple terminal usage. Most users at Berkeley choose *cs(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 *cs(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 sub-directory 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 these commands if you are a target.

If you use *cs(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

intro	introduction to commands
adb	debugger
addbib	create or extend bibliographic database
apply	apply a command to a set of arguments
apropos	locate commands by keyword lookup
ar	archive and library maintainer
as	VAX-11 assembler
at	execute commands at a later time
atq	print the queue of jobs waiting to be run
atrm	remove jobs spooled by at
awk	pattern scanning and processing language
basename	strip filename affixes
bc	arbitrary-precision arithmetic language
biff	be notified if mail arrives and who it is from
binmail	send or receive mail among users
cal	print calendar
calendar	reminder service
cat	catenate and print
cb	C program beautifier
cc	C compiler
ci	check in RCS revisions
cd	change working directory
checknr	check nroff/troff files
chfn	change password file information
chgrp	change group
chmod	change mode
chsh	change password file information
clear	clear terminal screen
cmp	compare two files
co	check out RCS revisions
col	filter reverse line feeds
colcrt	filter nroff output for CRT previewing
colrm	remove columns from a file
comm	select or reject lines common to two sorted files
compress	compress and expand data
cp	copy
crypt	encode/decode
csh	a shell (command interpreter) with C-like syntax
ctags	create a tags file
cvbackup	converts between VMS backup format and UNIX format
cvtnames	convert hashed file names
date	print and set the date
dbx	debugger
dc	desk calculator
dd	convert and copy a file
deroff	remove nroff, troff, tbl and eqn constructs
df	disk free
diction	print wordy sentences; thesaurus for diction
diff	differential file and directory comparator
diff3	3-way differential file comparison
du	summarize disk usage
echo	echo arguments
ed	text editor
efl	Extended Fortran Language
eqn	typeset mathematics

error analyze and disperse compiler error messages

eumlogin log into EUNICE process accounting

eumlogout log out of EUNICE process accounting

ex text editor

expand expand tabs to spaces, and vice versa

expr evaluate arguments as an expression

f77 Fortran 77 compiler

false provide truth values

file determine file type

filetype provides information about the file type

find find files

finger user information lookup program

fmt simple text formatter

fold fold long lines for finite width output device

fp Functional Programming language compiler/interpreter

fpr print Fortran file

from who is my mail from?

fsplit split a multi-routine Fortran file into individual files

ftp ARPANET file transfer program

gcore get core images of running processes

gprof display call graph profile data

graph draw a graph

grep search a file for a pattern

groups show group memberships

head give first few lines

hostid set or print identifier of current host system

hostname set or print name of current host system

ident identify files

indent indent and format C program source

install install binaries

iostat report I/O statistics

join relational database operator

kill terminate a process with extreme prejudice

last indicate last logins of users and teletypes

lastcomm show last commands executed in reverse order

ld link editor

learn computer aided instruction about UNIX

leave remind you when you have to leave

lex generator of lexical analysis programs

lint a C program verifier

lisp lisp interpreter

liszt compile a Franz Lisp program

ln make links

lock reserve a terminal

logger make entries in the system log

login sign on

look find lines in a sorted list

lookbib build inverted index for a bibliography, find references in a bibliography

lorder find ordering relation for an object library

lpq spool queue examination program

lpr off line print

lprm remove jobs from the line printer spooling queue

lptest generate lineprinter ripple pattern

ls list contents of directory

lxref lisp cross reference program

m4 macro processor

mailinfo tells the user that UNIX mail has been received

mail send and receive mail

make maintain program groups

man	find manual information by keywords; print out the manual
merge	three-way file merge
mesg	permit or deny messages
mkdir	make a directory
mkstr	create an error message file by massaging C source
more	file perusal filter for crt viewing
mset	retrieve ASCII to IBM 3270 keyboard map
msgsg	system messages and junk mail program
mt	magnetic tape manipulating program
mv	move or rename files
netstat	show network status
newaliases	rebuild the data base for the mail aliases file
nice	run a command at low priority (<i>sh</i> only)
nm	print name list
nroff	text formatting
od	octal, decimal, hex, ascii dump
pagesize	print system page size
passwd	change password file information
pc	Pascal compiler
pdx	pascal debugger
pi	Pascal interpreter code translator
pix	Pascal interpreter and executor
plot	graphics filters
pmerge	pascal file merger
pr	print file
printenv	print out the environment
prof	display profile data
ps	process status
ptx	permuted index
pwd	working directory name
px	Pascal interpreter
pxp	Pascal execution profiler
pxref	Pascal cross-reference program
quota	display disc usage and limits
ranlib	convert archives to random libraries
ratfor	rational Fortran dialect
rcp	remote file copy
rdist	remote file distribution program
rcs	change RCS file attributes
rcsdiff	compare RCS revisions
rcsmerge	merge RCS revisions
refer	find and insert literature references in documents
rev	reverse lines of a file
rlog	print log messages and other information about RCS files
rm	remove (unlink) files or directories
rmail	handle remote mail received via uucp
rmdir	remove (unlink) directories or files
roffbib	run off bibliographic database
rsh	remote shell
ruptime	show host status of local machines
rwho	who's logged in on local machines
sccs	front end for the SCCS subsystem
script	make typescript of terminal session
sed	stream editor
sendbug	mail a system bug report to 4bsd-bugs
sh	command language
size	size of an object file
sleep	suspend execution for an interval
soelim	eliminate .so's from nroff input

sort sort or merge files

sortbib sort bibliographic database

spell find spelling errors

spline interpolate smooth curve

split split a file into pieces

strings find the printable strings in a object, or other binary, file

strip remove symbols and relocation bits

struct structure Fortran programs

stty set terminal options

style analyze surface characteristics of a document

su substitute user id temporarily

sum sum and count blocks in a file

symorder rearrange name list

sysline display system status on status line of a terminal

sysstat display system statistics on a crt

tabs set terminal tabs

tail deliver the last part of a file

talk talk to another user

tar tape archiver

tbl format tables for nroff or troff

tc phototypesetter simulator

tcopy copy a mag tape

tee pipe fitting

telnet user interface to the TELNET protocol

test condition command

tftp trivial file transfer program

time time a command

tip connect to a remote system

tk paginator for the Tektronix 4014

m3270 full-screen remote login to IBM VM/CMS

touch update date last modified of a file

tp manipulate tape archive

tr translate characters

troff text formatting and typesetting

trpatch trace patch

true provide truth values

tset terminal dependent initialization

tsort topological sort

tty get terminal name

ul do underlining

unifdef remove ifdef'ed lines

uniq report repeated lines in a file

units conversion program

unixtovms switch UNIX file to VMS file format

uptime show how long system has been up

users compact list of users who are on the system

uucp unix to unix copy

uencode encode/decode a binary file for transmission via mail

uulog display UUCP log files

uname list names of UUCP hosts

uuq examine or manipulate the uucp queue

uuseid send a file to a remote host

uux unix to unix command execution

vacation return "I am on vacation" indication

version provides information about the EUNICE BSD version level

vgrind grind nice listings of programs

vi screen oriented (visual) display editor based on ex

vlp Format Lisp programs to be printed with nroff, vtroff, or troff

vms execute VMS commands from EUNICE shells

vmsas	VMS output assembler
vmsld	VMS link editor
vmsmail	send UNIX mail to the VMS mailer
vmsname	give equivalent VMS file specification
vmstat	report virtual memory statistics
vmstounix	switch VMS file to UNIX file format
vwidth	make troff width table for a font
w	who is on and what they are doing
wait	await completion of process
wall	write to all users
wc	word count
what	show what versions of object modules were used to construct a file
whatis	describe what a command is
whereis	locate source, binary, and or manual for program
which	locate a program file including aliases and paths (<i>cs/t</i> only)
who	who is on the system
whoami	print effective current user id
whois	DARPA Internet user name directory service
window	window environment
write	write to another user
xsend	secret mail
xstr	extract strings from C programs to implement shared strings
yacc	yet another compiler-compiler
yes	be repetitively affirmative

2. System Calls

intro	introduction to system calls and error numbers
accept	accept a connection on a socket
access	determine accessibility of file
acct	turn accounting on or off
adjtime	correct the time to allow synchronization of the system clock
bind	bind a name to a socket
brk	change data segment size
chdir	change current working directory
chmod	change mode of file
chown	change owner and group of a file
chroot	change root directory
close	delete a descriptor
connect	initiate a connection on a socket
creat	create a new file
dup	duplicate a descriptor
exec	execute a file
exit	terminate a process
fcntl	file control
flock	apply or remove an advisory lock on an open file
fork	create a new process
fsync	synchronize a file's in-core state with that on disk
getdtablesize	get descriptor table size
getgid	get group identity
getgroups	get group access list
gethostid	get/set unique identifier of current host
gethostname	get/set name of current host
getitimer	get/set value of interval timer
getpagesize	get system page size
getpeername	get name of connected peer
getpgrp	get process group
getpid	get process identification
getpriority	get/set program scheduling priority
getrlimit	control maximum system resource consumption

getrusage	get information about resource utilization
getsockname	get socket name
getsockopt	get and set options on sockets
gettimeofday	get/set date and time
getuid	get user identity
ioctl	control device
kill	send signal to a process
killpg	send signal to a process group
link	make a hard link to a file
listen	listen for connections on a socket
lseek	move read/write pointer
mkdir	make a directory file
mknod	make a special file
mount	mount or remove file system
open	open a file for reading or writing, or create a new file
pipe	create an interprocess communication channel
profil	execution time profile
ptrace	process trace
quota	manipulate disk quotas
read	read input
readlink	read value of a symbolic link
reboot	reboot system or halt processor
recv	receive a message from a socket
rename	change the name of a file
rmdir	remove a directory file
select	synchronous I/O multiplexing
send	send a message from a socket
setgroups	set group access list
setpgid	set process group
setquota	enable/disable quotas on a file system
setregid	set real and effective group ID
setreuid	set real and effective user ID's
shutdown	shut down part of a full-duplex connection
sigblock	block signals
sigpause	atomically release blocked signals and wait for interrupt
sigreturn	return from signal
sigsetmask	set current signal mask
sigstack	set and/or get signal stack context
sigvec	software signal facilities
socket	create an endpoint for communication
socketpair	create a pair of connected sockets
stat	get file status
swapon	add a swap device for interleaved paging/swapping
symlink	make symbolic link to a file
sync	update super-block
syscall	indirect system call
truncate	truncate a file to a specified length
umask	set file creation mode mask
unlink	remove directory entry
utimes	set file times
vfork	spawn new process in a virtual memory efficient way
vhangup	virtually "hangup" the current control terminal
wait	wait for process to terminate
write	write output

3. C Library Subroutines

intro	introduction to C library functions
abort	generate a fault
abs	integer absolute value

alarm	schedule signal after specified time
asinh	inverse hyperbolic functions
assert	program verification
atof	convert ASCII to numbers
bstring	bit and byte string operations
byteorder	convert values between host and network byte order
crypt	DES encryption
ctime	convert date and time to ASCII
ctype	character classification macros
cursor	screen functions with "optimal" cursor motion
dbm	data base subroutines
directory	directory operations
ecvt	output conversion
end	last locations in program
erf	error functions
execl	execute a file
exit	terminate a process after flushing any pending output
exp	exponential, logarithm, power
fclose	close or flush a stream
ferror	stream status inquiries
floor	absolute value, floor, ceiling, and round-to-nearest functions
fopen	open a stream
fread	buffered binary input/output
frexp	split into mantissa and exponent
fseek	reposition a stream
getc	get character or word from stream
getdiskbyname	get disk description by its name
getenv	value for environment name
getfsent	get file system descriptor file entry
getgrent	get group file entry
gethostbyname	get network host entry
getlogin	get login name
getnetent	get network entry
getopt	get option letter from argv
getpass	read a password
getprotoent	get protocol entry
getpw	get name from uid
getpwent	get password file entry
gets	get a string from a stream
getservent	get service entry
gettyent	get tty's file entry
getusershell	get legal user shells
getwd	get current working directory pathname
hypot	Euclidean distance, complex absolute value
ieee	copysign, remainder, exponent manipulations
inet	Internet address manipulation routines
infnan	signals invalid floating-point operations on a VAX (temporary)
initgroups	initialize group access list
insque	insert/remove element from a queue
j0	bessel functions
lgamma	log gamma function
lib2648	subroutines for the HP 2648 graphics terminal
malloc	memory allocator
math	introduction to mathematical library functions
mktemp	make a unique file name
monitor	prepare execution profile
mp	multiple precision integer arithmetic
ndbm	data base subroutines
nice	set program priority

nlist get entries from name list

ns Xerox NS(tm) address conversion routines

pause stop until signal

perror system error messages

plot graphics interface

popen initiate I/O to/from a process

printf formatted output conversion

psignal system signal messages

putc put character or word on a stream

puts put a string on a stream

qsort quicker sort

rand random number generator

random better random number generator; routines for changing generators

rcmd routines for returning a stream to a remote command

regex regular expression handler

resolver resolver routines

rexec return stream to a remote command

scandir scan a directory

scanf formatted input conversion

setbuf assign buffering to a stream

setjmp non-local goto

setuid set user and group ID

siginterrupt allow signals to interrupt system calls

signal simplified software signal facilities

sin trigonometric functions and their inverses

sinh hyperbolic functions

sleep suspend execution for interval

sqrt cube root, square root

stdio standard buffered input/output package

string string operations

stty set and get terminal state (defunct)

swab swap bytes

syslog control system log

system issue a shell command

termcap terminal independent operation routines

time get date and time

times get process times

ttyname find name of a terminal

ualarm schedule signal after specified time

ungetc push character back into input stream

usleep suspend execution for interval

utime set file times

valloc aligned memory allocator

varargs variable argument list

vlimit control maximum system resource consumption

vtimes get information about resource utilization

3f. Fortran Library

intro introduction to FORTRAN library functions

abort abnormal termination

access determine accessibility of a file

alarm execute a subroutine after a specified time

bessel of two kinds for integer orders

bit and, or, xor, not, rshift, lshift bitwise functions

chdir change default directory

chmod change mode of a file

etime return elapsed execution time

exit terminate process with status

fdate return date and time in an ASCII string

fmin	return extreme values
flush	flush output to a logical unit
fork	create a copy of this process
fseek	reposition a file on a logical unit
getarg	return command line arguments
getc	get a character from a logical unit
getcwd	get pathname of current working directory
getenv	get value of environment variables
getlog	get user's login name
getpid	get process id
getuid	get user or group ID of the caller
hostnm	get name of current host
idate	return date or time in numerical form
index	tell about character objects
ioinit	change f77 I/O initialization
kill	send a signal to a process
link	make a link to an existing file
loc	return the address of an object
long	integer object conversion
malloc	memory allocator
perror	get system error messages
plot	f77 library interface to <i>plot</i> (3X) libraries.
putc	write a character to a fortran logical unit
qsort	quick sort
rand	return random values
random	better random number generator
rename	rename a file
signal	change the action for a signal
sleep	suspend execution for an interval
stat	get file status
system	execute a UNIX command
time	return system time
topen	f77 tape I/O
traper	trap arithmetic errors
trapov	trap and repair floating point overflow
trpfe	trap and repair floating point faults
tynam	find name of a terminal port
unlink	remove a directory entry
wait	wait for a process to terminate

4. Special Files

intro	introduction to special files and hardware support
cons	VAX-11 console interface
fl	console floppy interface
mt	TM78/TU-78 MASSBUS magtape interface
null	data sink
printer	line printer interface
rta	DECnet virtual terminal interface
ty	general terminal interface
ttyp	tcp/ip virtual terminal interface

5. File Formats

intro	file formats and conventions
L-devices	UUCP device description file
L-dialcodes	UUCP phone number index file
L.aliases	UUCP hostname alias file
L.cmds	UUCP remote command permissions file
L.sys	UUCP remote host description file

USERFILE	UUCP pathname permissions file
a.out	assembler and link editor output
acct	execution accounting file
aliases	aliases file for sendmail
ar	archive (library) file format
core	format of memory image file
dbx	dbx symbol table information
dir	format of directories
disktab	disk description file
dump	incremental dump format
fs	format of file system volume
fstab	static information about the filesystems
gettytab	terminal configuration data base
group	group file
hosts	host name data base
map3270	database for mapping ascii keystrokes into IBM 3270 keys
mtab	mounted file system table
networks	network name data base
passwd	password file
phones	remote host phone number data base
plot	graphics interface
printcap	printer capability data base
protocols	protocol name data base
rcsfile	format of RCS file
remote	remote host description file
resolver	resolver configuration file
services	service name data base
stab	symbol table types
tar	tape archive file format
termcap	terminal capability data base
tp	DEC/mag tape formats
ttys	terminal initialization data
types	primitive system data types
utmp	login records
uuencode	format of an encoded uuencode file
vfont	font formats for the Benson-Varian or Versatec
vgrindfs	vgrind's language definition data base

6. Games

intro	programs designed solely for entertainment
aardvark	yet another exploration game
adventure	an exploration game
arithmetic	provide drill in number facts
backgammon	the game
banner	print large banner on printer
battlestar	a tropical adventure game
bcd	convert to antique media
boggle	play the game of boggle
canfield	the solitaire card game canfield
chess	the game of chess
ching	the book of changes and other cookies
cribbage	the card game cribbage
doctor	interact with a psychoanalyst
fish	play "Go Fish"
fortune	print a random, hopefully interesting, adage
hangman	Computer version of the game hangman
hunt	a multi-player multi-terminal game
mille	play Mille Bourmes
monop	Monopoly game

number	convert Arabic numerals to English
quiz	test your knowledge
rain	animated raindrops display
robots	fight off villainous robots
rogue	Exploring The Dungeons of Doom
sail	multi-user wooden ships and iron men
snake	display chase game
trek	trekkie game
worm	Play the growing worm game
worms	animate worms on a display terminal
wump	the game of hunt-the-wumpus
zork	the game of dungeon

7. Miscellaneous

intro	miscellaneous useful information pages
ascii	map of ASCII character set
environ	user environment
eqnchar	special character definitions for eqn
greek	graphics for extended TTY-37 type-box
hier	file system hierarchy
mailaddr	mail addressing description
man	macros to typeset manual
me	macros for formatting papers
ms	text formatting macros
term	conventional names for terminals

8. System Maintenance

intro	introduction to system maintenance and operation commands
XNSrouted	not currently implemented in EUNICE
ac	not currently implemented in EUNICE
adduser	not currently implemented in EUNICE
arff	not currently implemented in EUNICE
arp	not currently implemented in EUNICE
bad144	not currently implemented in EUNICE
badsect	not currently implemented in EUNICE
bugfiler	not currently implemented in EUNICE
catman	create the cat files for the manual
chown	change owner
clri	not currently implemented in EUNICE
comsat	not currently implemented in EUNICE
config	not currently implemented in EUNICE
crash	not currently implemented in EUNICE
cron	clock daemon
cvtuaf	convert sysuaf file to /etc/password
dcheck	not currently implemented in EUNICE
diskpart	not currently implemented in EUNICE
dmesg	not currently implemented in EUNICE
drtest	not currently implemented in EUNICE
dump	not currently implemented in EUNICE
dumpfs	not currently implemented in EUNICE
edquota	not currently implemented in EUNICE
fastboot	not currently implemented in EUNICE
fingerd	not currently implemented in EUNICE
format	not currently implemented in EUNICE
fsck	not currently implemented in EUNICE
ftpd	not currently implemented in EUNICE
gettable	not currently implemented in EUNICE
getty	not currently implemented in EUNICE

halt	not currently implemented in EUNICE
htable	not currently implemented in EUNICE
icheck	not currently implemented in EUNICE
ifconfig	not currently implemented in EUNICE
implog	not currently implemented in EUNICE
implogd	not currently implemented in EUNICE
inetd	not currently implemented in EUNICE
init	not currently implemented in EUNICE
kgmon	not currently implemented in EUNICE
lpc	not currently implemented in EUNICE
lpd	not currently implemented in EUNICE
makedev	not currently implemented in EUNICE
makekey	generate encryption key
mketcgrp	make /etc/group
mkfs	not currently implemented in EUNICE
mkhosts	not currently implemented in EUNICE
mklost+found	not currently implemented in EUNICE
mknod	not currently implemented in EUNICE
mkpasswd	generate hashed password table
mkproto	not currently implemented in EUNICE
mount	not currently implemented in EUNICE
named	not currently implemented in EUNICE
ncheck	not currently implemented in EUNICE
newfs	not currently implemented in EUNICE
pac	not currently implemented in EUNICE
ping	not currently implemented in EUNICE
prtusers	provide information about EUNICE process accounting
pstat	not currently implemented in EUNICE
quot	not currently implemented in EUNICE
quotacheck	not currently implemented in EUNICE
quotaon	not currently implemented in EUNICE
rc	not currently implemented in EUNICE
rdump	not currently implemented in EUNICE
reboot	not currently implemented in EUNICE
renice	not currently implemented in EUNICE
repquota	not currently implemented in EUNICE
restore	not currently implemented in EUNICE
rexeed	not currently implemented in EUNICE
rlogind	not currently implemented in EUNICE
rmt	not currently implemented in EUNICE
route	not currently implemented in EUNICE
routed	not currently implemented in EUNICE
rrestore	not currently implemented in EUNICE
rshd	not currently implemented in EUNICE
rwhod	not currently implemented in EUNICE
rxformat	not currently implemented in EUNICE
sa	not currently implemented in EUNICE
savecore	not currently implemented in EUNICE
sendmail	send mail over the internet
shutdown	not currently implemented in EUNICE
slattach	not currently implemented in EUNICE
sticky	not currently implemented in EUNICE
swapon	not currently implemented in EUNICE
sync	not currently implemented in EUNICE
syslogd	not currently implemented in EUNICE
talkd	not currently implemented in EUNICE
telnetd	not currently implemented in EUNICE
tftpd	not currently implemented in EUNICE
timed	not currently implemented in EUNICE

timedc	not currently implemented in EUNICE
trpt	not currently implemented in EUNICE
trsp	not currently implemented in EUNICE
tunefs	not currently implemented in EUNICE
update	not currently implemented in EUNICE
uucico	transfer files queued by uucp or uux
uuclean	uucp spool directory clean-up
uupoll	poll a remote UUCP site
uusnap	show snapshot of the UUCP system
uuxqt	UUCP execution file interpreter
vipw	edit the password file