5ESS® Switch
UNIX RTR Operating System Reference Manual

Document: 235-700-200
Issue Date: December 2001
Issue Number: 10.00

Ordering Information

This information product is distributed by the Lucent Technologies Customer Information Center in

Indianapolis, Indiana.

The order number for this information product is 235-700-200. To order, call the following:

1-888-LUCENT8 (1-888-582-3688) or fax to 1-800-566-9568 (from inside the continental United States)

+1-317-322-6847 or fax to +1-317-322-6699 (from outside the continental United States).

Support Telephone Numbers

Information Product Support Telephone Number: To report errors or ask nontechnical questions about this or other information products produced by Lucent Technologies, call 1-800-645-6759.

Technical Support Telephone Numbers: For initial technical assistance, call the North American Regional Technical Assistance Center (NARTAC) at 1-800-225-RTAC (1-800-225-7822). For further assistance, call the Customer Technical Assistance Management (CTAM) center as follows:

1-800-225-4672 (from inside the continental United States)

+1-630-224-4672 (from outside the continental United States).

Both centers are staffed 24 hours a day, 7 days a week.

Acknowledgment

Developed by Lucent Technologies Customer Training and Information Products.

Comment Form

Lucent Technologies values your comments!

Lucent Technologies welcomes your comments on this information product. Your opinion is of great value and helps us to improve. Please print out this form and complete it. Please fax the form to 407 767 2760 (U.S.) or +1 407 767 2760 (outside the U.S.). Or, you may email comments to: ctiphotline@lucent.com

Product Line: 5ESS Switch
Title:
UNIX RTR Operating System Reference
Information Product Code: 235-700-200
Issue Number: 10.00
Publication Date: December 2001

(1) Was the information product:

	Yes	**No**	**Not Applicable**
In the language of your choice?			
In the desired media (paper, CD-ROM, etc.)?			
Available when you needed it?			

Please provide any additional comments:

(2) Please rate the effectiveness of the information product:

	Excellent	**More than satisfactory**	**Satisfactory**	**Less than satisfactory**	**Unsatisfactory**	**Not applicable**
Ease of use						
Level of detail						
Readability and clarity						
Organization						
Completeness						
Technical accuracy						
Quality of translation						
Appearance						

If your response to any of the above questions is "Less than satisfactory" or "Unsatisfactory", please explain your rating.

(3) If you could change one thing about this information product, what would it be?

(4) Please write any other comments about this information product:

. . . .

Please complete the following if we may contact you for clarification or to address your concerns:

Name: ____

Company/organization: _____________________________________

Telephone number: ____

Address: ____

____

Email Address: ____

Job function: ____

1. INTRODUCTION

1.1 PURPOSE

This manual is intended for personnel using the *UNIX*® Real-Time Reliable (RTR) operating system features of the Lucent 3B20D or 3B21D computer in the *5ESS*® switch. This manual provides the *UNIX*® RTR operating system available commands and instructions for setting up logins for the 5E13 and later software releases.

IT IS ASSUMED THAT THE USER OF THIS MANUAL KNOWS HOW TO USE THE*UNIX*® RTR OPERATING SYSTEM. For users who *do not* know how to use the *UNIX*® RTR operating system, formal training is available through a Lucent technical training location. Contact your account representative for more information.

WARNING: Use the commands in Section 5 of this manual as printed. Any deviations or misuse of these procedures CAN result in a *5ESS*® switch failure.

1.2 UPDATE INFORMATION

1.2.1 REASON FOR UPDATE

This is an update to the *5ESS*® *Switch UNIX*® *RTR Operating System Reference Manual* 235-700-200, Issue 9.00 dated November 2000. This update covers software releases 5E13 and later software releases.

The 5E15 FR4 software release adds the *loadmccpasswd* command.

This update also includes editorial changes.

1.2.2 SUPPORTED SOFTWARE RELEASES

In accordance with the 5ESS Switch Software Support Product Plan, the 5E12 software release is rated Discontinued Availability (DA) as of September 2000. The information supporting 5E12 and earlier software releases is being removed over time, instead of concurrently, from all documentation.

If you are supporting offices that use a software release prior to 5E13 and you have a need for the information that is being removed, retain the associated pages as they are removed from the paper documents, or retain the earlier copy of the CD-ROM.

1.2.3 TERMINOLOGY

1.2.3.1 Communication Module Name Change

The term Communication Module (CM) has been changed to the Global Messaging Server (GMS), representing the new portfolio name of this particular module. The current names of the specific types of the GMS (the CM2 and CM3) have not been changed. Where the CM name has been used in a generic way within this information product, the name will be changed to GMS. Where the specific version of GMS (CM2 or CM3) is being described or mentioned, the name will not be changed. However, the GMS name may be added to the description in certain places as a reminder of the change, and that the particular version is a part of the overall portfolio. The following examples show how these names may be used together:

- Global Messaging Server (formerly Communication Module)
- GMS (formerly CM)

- Global Messaging Server-CM2
- GMS-CM2
- Global Messaging Server-CM3
- GMS-CM3.

These name changes will be made over time as other technical changes are required. Also, these changes may not be reflected in all software interfaces (input and output messages, master control center screens, and recent change and verify screens). Where the document references these areas, the names are used as they are within the software interface.

1.2.3.2 5ESS®-2000 Switch Name Change

This *5ESS* switch document may contain references to the *5ESS* switch, the *5ESS*-2000 switch, the *5ESS* AnyMedia® Switch, and the *7R/E*™ *5ESS* switch. The official name of the product has been changed back to the *5ESS* switch. In the interim, assume that any reference, in this document, to the *5ESS*-2000 switch, the *5ESS* AnyMedia Switch, or the 7R/E *5ESS* switch is also applicable to the *5ESS* switch. It should be noted that this name change may not have been carried forward into software-influenced items such as input and output messages, master control center screens, and recent/change verify screens.

1.2.3.3 Document Specific Terminology

All acronyms, abbreviations, and any software or document specific terms are defined in the GLOSSARY, located in the back of this document.

1.3 ORGANIZATION

This manual is divided into the following sections separated by tabs.

- Introduction
- Getting Started
- Administration
- EMACS Description
- Commands
- Glossary
- Index.

The *Introduction* describes what this manual contains and the *UNIX*® RTR Operating System Availability feature.

The *Getting Started* section describes the following:

- Logging in
- Logging out
- Communicating through your terminal
- Running a program

- The current directory
- Pathnames
- Surprises.

The *Administration* section is intended for use by an administrator. This section provides procedures for securing the dial-up access to a supplementary trunk line work station or a text recent change terminal using new login software.

The *EMACS Description* section describes the EMACS screen editor and provides the available commands and options.

The *Commands* section describes the available *UNIX*® RTR operating system commands and programs. These programs are invoked directly by the user or by command language procedures, as opposed to subroutines, which are invoked by the user's programs. Commands generally reside in the directory **/bin** (for **bin**ary programs). Some programs also reside in **/usr/bin**, to save space in **/bin**. The directories are also searched automatically by the command interpreter called the *shell*. Commands that reside in other directories are noted on the command page.

This section consists of many independent entries that are entered into the documentation in alphabetical order. The name of the entry appears in the first line of text of the page. Some entries may describe several routines, commands, etc. In such cases, the entry appears only once, alphabetized under its "major" name.

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

- **NAME** gives the name(s) of the entry and briefly states its purpose.
- **SYNOPSIS** summarizes the use of the program being described.
- **DESCRIPTION** provides additional information about the program or facility outlined in "Name" and "Synopsis" .
- **EXAMPLES** gives example(s) of usage, where appropriate.
- **FILES** gives the filenames that are built into the program.
- **SEE ALSO** refers to related information.
- **DIAGNOSTICS** discusses the diagnostic indications that may be produced. Messages that are intended to be self-explanatory are not listed.
- **WARNINGS** provides potential pitfalls.
- **BUGS** provides known bugs and sometimes deficiencies. Occasionally, the suggested fix is also described.

The following *conventions* are used in this section:

- **Boldface** strings are literals and are to be typed just as they appear.
- *Italic* strings usually represent variables and program names found elsewhere in the manual. Note that this convention is not used in the "SYNOPSIS" or "SEE ALSO" parts; regular print is used in place of italics.
- Square brackets **[ ]** around an argument prototype show that the argument is optional. When an

argument prototype is given as "name" or "file", 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 (**-**), plus (**+**), or equal sign (**=**) is often taken to be some sort of flag argument, even if it appears in a position where a filename could appear. Therefore, it is not wise to have files whose names begin with **-**, **+**, or **=**.

On most systems, all entries are available on line by using the *man* command. As these on-line entries are updated from time to time, the on-line version may disagree with the entries in this manual. When they do disagree, the on-line version is the correct version to use.

1.4 USER FEEDBACK

We are constantly striving to improve the quality and usability of this information product. Please use one of the following options to provide us with your comments:

- You may use the on-line comment form at **http://www.lucent-info.com/comments**
- You may email your comments to **ctiphotline@lucent.com**
- You may print the comment form (located at the beginning of this information product after the Legal Notice) to send us your comments either by fax or mail as follows:
  - Fax to **1-407-767-2760**
  - Mail to the following address:

    Lucent Technologies
    Documentation Services Coordinator
    240 E. Central Parkway
    Altamonte Springs, FL 32701-9928

- You may call the hot line with your comments. The telephone number is **1-800-645-6759**. The hot line is staffed Monday through Friday from 8:00 a.m. to 5:00 p.m. Eastern time.

Please include with your comments the title, ordering number, issue number, and issue date of the information product, your complete mailing address, and your telephone number.

If you have questions or comments about the distribution of our information products, see Section 1.5 , Distribution.

1.5 DISTRIBUTION

For distribution comments or questions, either contact your local Lucent Technologies Account Representative or send them directly to the Lucent Technologies Customer Information Center (CIC) in Indianapolis, Indiana.

A documentation coordinator has authorization from Lucent Technologies to purchase our information products at discounted prices. To find out whether your company has this authorization through a documentation coordinator, call **1-888-LUCENT8 (1-888-582-3688)**.

Customers who are not represented by a documentation coordinator and employees of Lucent Technologies should order *5ESS*® switch information products directly from CIC.

To order, call the following telephone number:

- **1-888-LUCENT8 (1-888-582-3688)** or fax to **1-800-566-9568** (from inside the continental United States)
- **+1-317-322-6847** or fax to **+1-317-322-6699** (from outside the continental United States).

1.6 TECHNICAL ASSISTANCE

For initial technical assistance, call the North American Regional Technical Assistance Center (NARTAC) at **1-800-225-RTAC (1-800-225-7822)**.

For further assistance, call the Customer Technical Assistance Management (CTAM) center at the following number:

- **1-800-225-4672** (from inside the continental United States)
- **+1-630-224-4672** (from outside the continental United States).

Both centers are staffed 24 hours a day, 7 days a week.

1.7 *UNIX*® RTR OPERATING SYSTEM AVAILABILITY FEATURE

1.7.1 DESCRIPTION

The *UNIX*® RTR Operating System Availability feature allows users to access the spare computing power of the 3B20D computer for small administrative tasks. For example, users could maintain work schedules and spare circuit pack inventory lists on the 3B20D computer.

The *UNIX*® RTR Operating System Availability feature provides the following:

- Two disk partitions; *unixa* and *unixabf*. The *unixa* partition is mounted as */unixa*. This partition contains the on-line manual pages and the control, and spooling files for the *at*, *cron*, and *lpr* commands. The *unixabf* partition is mounted as */unixa/users*. This partition provides an area for users to create files for their own use. The system is designed so that excess file usage in these partitions does not overflow into the partitions needed for switch operation. In small disk configurations, the *unixabf* partition is not available. In these configurations, the *unixa* partition is also used for user-created files.
- The*UNIX*® RTR operating system command, *admin*, that allows the creation of nonroot logins. It is **strongly** advised that the root login only be used under direction of Lucent field support.
- The *UNIX*® RTR operating system command, *lpr*, to spool output to the ROP (receive-only printer).
- On-line manual pages and a *UNIX*® RTR operating system command, *man*, to access them.
- The *UNIX*® RTR operating system commands, *at* and *cron*, to allow scheduling commands in the future.

1.7.2 FILE ORGANIZATION

The */unixa* or */unixa/users* directory is available for users to create and store files. The */unixa* directory is structured as follows:

```
/unixa/users/manager            # HOME directory for manager
```

```
|    /al                 # HOME directory for al

|    /don                  # HOME directory for don

|    /jim                 # HOME directory for jim

|

/man/1.admin              # |

|  /...                  # |On-line manual pages

|  /1.write              # |

|

/tmp                       # Directory to hold temporary files

|

/spool/cron/crontabs/root    # Root's crontab file

           |       /don     # Don's crontab file

           |

           /atjobs/b.123456  # Batch job control file

                  /b.6780    # Batch job control file

                  /a.980     # At job control file
```

Unless directed by Lucent field support, create files in (or below) your *HOME* directory or the */unixa/tmp* directory. Do not delete or change any files outside these directories.

2. GETTING STARTED

2.1 INTRODUCTION

This section provides the following basic information needed to get started:

- Logging in to the *UNIX*® RTR operating system
- Communicating through a terminal
- Running a program
- The current directory
- Pathnames
- Surprises.

Before using any commands and procedures on the *5ESS* switch, read this manual thoroughly, and use only the commands and procedures provided in this manual. Additional information about the *UNIX*® commands is available commercially.

2.2 LOGGING IN

It is important that the login name be typed in lowercase, if possible; if the login name is typed in uppercase letters, the system assumes that the terminal cannot generate lowercase letters and that all further uppercase input is to be treated as lowercase.

Most terminals have a speed switch that should be set to the appropriate speed and a half-/full-duplex switch that should be set to full-duplex. When a connection is established, the system displays **login:**. Complete the following steps to log in to the system:

(1) At the **login:** prompt, enter the respective username and press Enter.

Response: The system prompts for the user **PASSWORD**.

(2) Enter the respective password and press Enter. The password does not print (echo) on the terminal display.

Response: The system returns a **$** prompt indicating a successful login.

The shell is described in the Section 2.4 , "Running a Program."

For detailed information about the login sequence, refer to *login*. For information about describing the characteristics of your terminal to the system, refer to *stty*. *Profile* explains how to accomplish this task automatically every time you log in. For information about the logout procedure, see Section 3.5.2 .

2.3 COMMUNICATING THROUGH YOUR TERMINAL

When input is typed, the system gathers the characters and saves them. These characters are not given to a program until "Enter" or "Newline" is pressed.

The terminal is a full-duplex input/output terminal with full read-ahead capability. Full read-ahead capability allows typing at any time, even while a program is prompting the user. If typing is done during output, the output is interspersed with the input characters; however, the input is saved and interpreted in the correct sequence. There is a limit to the amount of read-ahead, but it is generous and not likely to be exceeded unless the system is in trouble. When the read-ahead limit is exceeded, the system throws away *all* the saved characters.

On an input line from a terminal, the character @ "kills" all the characters typed before it. The character # erases the last character typed. Successive uses of # erase characters back to, but not beyond, the beginning of the line; @ and # can be typed as themselves by preceding them with \; thus, to erase a \, two #s are needed. These default erase and kill characters can be changed; see *stty*.

The ASCII **DC3** (CONTROL-S) character can be used to temporarily stop output. It is useful with cathode ray terminals (CRT) to prevent output from disappearing before it can be read. Output is resumed when a **DC1** (CONTROL-Q) or a second **DC3** (or any other character, for that matter) is typed. The **DC1** and **DC3** characters are not passed to any other program when used in this manner.

The ASCII **DEL** (also known as rubout) character is not passed to programs but instead generates an interrupt signal, just like the "break," "interrupt," or "attention" signal. This signal, treated as lowercase, generally causes a running program to stop. It is typically used to stop a long printout that is not wanted. However, programs can arrange either to ignore this signal altogether or to be notified when it happens (instead of being stopped). The editor *ed*, for example, catches interrupts and stops what it is doing, instead of terminating, so that an interrupt can be used to halt an editor printout without losing the file being edited.

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

Besides adapting to the speed of the terminal, the *UNIX*® RTR operating system tries to determine whether the terminal of the user has the "new-line" function, or whether it must be simulated with a "carriage-return" and "line-feed" pair. In the latter case, all *input* "carriage-return" characters are changed to "line-feed" characters (the standard line delimiter); and a "carriage-return" and "line-feed" pair is echoed to the terminal. If you get into the wrong mode, use the *stty* command to correct it.

Tab characters are used freely in system source programs. If the terminal of the user does not have the tab function, tab characters may be changed into spaces during output and echoed as spaces during input. The *stty* command can be used to set or reset this mode. The system assumes that tabs are set every eight-character positions.

2.4 RUNNING A PROGRAM

When a successful login is established, the shell program listens to the terminal. The shell program reads the lines typed, splits them into a command name and its arguments, and executes the command. A command is an executable program. Normally, the shell looks in the current directory first (see Section 2.5 "The Current Directory") for a program with the given name. If the program is not in the current directory, the shell looks in the system directories. The system-provided commands are kept in directories where the shell can find them. Commands can also be kept in user directories, and the shell can be arranged to find them there.

The command name is the first word on an input line to the shell. The command and its arguments are separated from one another by space and/or tab characters.

When a program stops, the shell regains control and returns a **$** prompt to show that it is ready for another command. For more detailed information about shell capabilities, refer to *sh*.

2.5 THE CURRENT DIRECTORY

The *UNIX*® RTR operating system file system is arranged in an hierarchy of directories. When the system administrator provides a user name, a directory is also created for the user (ordinarily with the same name as the user name). This directory is known as the *login* **or** *home* directory. When the user logs in, the home directory becomes the *current* or *working* directory; and any file created is, by default, assumed to be in that directory. Because the user is the owner of this directory, the user has full permissions to read, write, alter, or destroy the contents of the home directory. Permissions for directories and files not owned by a user may be granted or denied by their owners or by the system administrator. To change the current directory, use *cd* [explained under *sh*(1)].

2.6 PATHNAMES

To refer to files not in the current directory, a pathname must be used. Full pathnames begin with "**/**", which is the *root* directory of the file system. Following the initial slash, each directory and subdirectory is named in succession until the filename is reached. Each directory name is followed by a "**/**". For example, **/usr/ae/filex** refers to file **filex** in directory **ae**; **ae** is a subdirectory of **usr**; **usr** is in the root directory.

If your current directory contains subdirectories, the pathnames of files therein begin with the name of the corresponding subdirectory (*without* a prefixed **/**). Without important exception, a pathname may be used anywhere a filename is required.

Important commands that change the contents of files are *cp*, *mv*, and *rm*, which copy, move (that is, rename), and remove files. To find out the status of files or directories, use *ls*. Use *mkdir* for making directories and *rmdir* for removing directories.

2.7 SURPRISES

Certain commands provide inter-user communication. It is important that users become familiar with these commands even if the user does not plan to use them initially. To communicate with another user currently logged in, *write* or *mail* is used. The *mail* command leaves a message notifying the user has mail when the user logs in.

When logging in, a message-of-the-day may be presented before the first **$** prompt.

3. ADMINISTRATION

3.1 INTRODUCTION

3.1.1 DIAL-UP FACILITIES

Using dial-up facilities such as the dial-up supplementary trunk and line work station (STLWS) and the dial-up text recent change terminal greatly enhances the effectiveness of the field support effort. However, if data sets are left powered-on, unauthorized access to a *5ESS*® switch is possible. For convenience, the data sets are usually left powered-on.

This section describes the procedures for securing the dial-up access to an STLWS or a text recent change terminal using new login software available with the *UNIX*® RTR operating system in 5E13 and later software release offices.

3.1.2 LOGIN CAPABILITY FEATURES

The login software provides login and password security for the *UNIX*® RTR operating system user environment. This section provides procedures for adding a login to the STLWS or the text recent change terminal; however, any terminal [except the master control center (MCC) and remote maintenance facilities] can be modified to require a login by making the necessary equipment configuration data changes.

The login code provides some additional flexibility in the configuration of the terminals. For users requiring only *UNIX*® RTR operating system capability, the login software allows the STLWS and the recent change terminals to be logged in directly as *UNIX*® RTR operating system terminals.

This section also describes an optional second password (similar to the External Security Code or Network Access Password used on other systems) which may be required for all logins.

3.2 HARDWARE

The only hardware concern for the login capability is the cable from the input/output processor (IOP) to the data set. This cable is different from a standard IOP cable used with terminals in the office. The standard IOP cable, designed for use with terminals, will not work with a data set. A diagram of the correct cable is shown in Figure 3-1 . The correct cable includes all necessary connections to allow the data set to signal the IOP when a user hangs up the telephone. If no login is required, this signaling is not necessary. If the port requires a login and an incorrect cable is used, only the first user after each teletype restore, is required to login.

3.3 SOFTWARE

3.3.1 GENERAL

Software changes for the login capability include updating the */etc/passwd* file to establish the login and recent change modifications to execute the login process.

3.3.2 ADDING NEW LOGINS

New logins are added by updating the file */etc/passwd*. The user must be in the *UNIX*® RTR operating system to add new logins. Enter the *UNIX*® RTR operating system with the command:

rcv:menu,sh;

Adding new logins is done by using the *admin* tool available on *UNIX*® RTR operating systems. [See the manual page for *admin* in the Commands section of this manual.]

Admin makes all required entries in the */etc/passwd* file while protecting other entries from possible corruption. *Admin* does not provide a password. The user is prompted for a password when the login is first used.

To secure the dial-up terminals, the following logins may be established using *admin*:

(1) To set up a *UNIX*® RTR operating system login, enter the following command:

admin -a *login_name* **-u**

(2) To set up a Craft shell login, enter the following command:

admin -a *login_name* **-p**

In the preceding examples, the **-a** option is the login name; the **-u** option shows that this is a *UNIX*® RTR operating system login; and the **-p** option shows that this login uses a *pds/mml* shell.

Note that *login_name* is a variable that is replaced by the actual login.

Printing the */etc/passwd* shows the new logins. A description of an entry in */etc/passwd* is shown in Figure 3-2 .

3.3.3 ADDING A SECOND PASSWORD (OPTIONAL)

If additional security is desired, a second password may be added. To add the second password, enter the following command:

passwd deamon

The user is prompted to enter the second password.

Dial-up users are then prompted for the "Machine Password" when they log on. The deamon password is entered in response.

Note there is only one "Machine Password" for all logins. This provides an effective method of securing all logins without changing individual passwords.

To clear the deamon password, enter the following command:

admin -c deamon

This command removes the deamon password, and future users are not required to enter a password.

When the new logins and the optional second password have been entered, exit the *UNIX*® RTR operating system by entering a CONTROL-D.

NOTE: Procedures in Sections 3.3.1 through 3.3.3 , though still supported, are not recommended with the advent of the *5ESS*® Switch Command Restriction procedure. The preferred login and password administration scheme is *authority management*. Authority management is documented in the 235-105-210, *Routine Operations and Maintenance,*.

3.3.4 DATABASE

3.3.4.1 General

Database changes are required to convert an STLWS or recent change terminal into a secure dial-up facility. It is assumed that the user is familiar with the use of recent change and with equipping these dial-up facilities. The following procedures assume that the teletypewriter controller and the teletypewriter are equipped and that all hardware has been connected to the proper location on the IOP backplane.

3.3.4.2 Database Changes for a Dial-Up STLWS

Perform following ECD changes to set up a secure dial-up STLWS terminal:

Lower the Baud Rate Lowering the STLWS baud rate to 1200 baud requires one change on the **ciopt** form. If the terminal is on port /dev/tty**xx**, then the correct **ciopt** form to select is ttyop**xx**. For example, if the terminal is on port /dev/tty9, select **ciopt** form ttyop9, /dev/tty10 select ttyop10, etc. On this form, change ttopt_name (field 2) to **PDS12** (black and white terminal) or **PDS12C** (color terminal).

Large Scroll Region: On Page 4 of the **cdopt** forms STLWSCG (color terminal) and VT100DAP (black and white terminal), change field 60, sub-fields 11 and 12, from
'2', '3' to '0', '2'. Change field 94, first item from '23' to '2', also.

Set Up Paths for the Login Process To execute the login process on an STLWS terminal, ECD changes are required on two **getty** forms. Two forms are involved since one form is required for the message section of the STLWS screen and a second form is used by the display region of the screen. The two "getty" form names differ only in that the TTY name in gettyres (field 1) is lowercase on one form and uppercase on the other form; for example: **gettyl** and **gettyL**. The same change is required on both **getty** forms. Change shlname (field 4) to **/etc/login**.

3.3.4.3 Database Changes for a Dial-Up Recent Change Terminal

Perform following ECD changes to set up a secure dial-up recent change terminal.

Lower the Baud Rate Lowering the recent change terminal baud rate to 1200 baud may require one change on the **ciopt** form. If your terminal is on port /dev/tty**xx**, then the correct **ciopt** form to select is ttyop**xx**. For example, if your terminal is on port /dev/tty9, select **ciopt** form ttyop9, /dev/tty10 select ttyop10, etc. On this form, change ttopt_name (field 2) to **PDS12**.

Set Up the Path for the Login Process To execute the login process for a dial-up recent change terminal, an ECD change is required on one **getty** form. The TTY name in gettyrec (field 1) is lowercase (for example: **gettyw**). Since this is a text recent change terminal and there is no display region, only one form requires updating.

On the **getty** form, change shlname (field 4) to **/etc/login**.

3.3.4.4 Computer Access Restriction

For increased security, the Computer Access Restriction (CAR) feature may also be used. This feature restricts access to your modem line(s) to only callers from a listed set of telephone numbers.

Instructions for setting up the CAR feature can be found in document 235-190-130 *Local Area Signaling Services (LASS) User's Manual*, Section 9.

3.4 INITIALIZE THE TELETYPEWRITER CONTROLLER

After the ECD changes have been completed, the new options must be down loaded into the teletypewriter controller (TTYC) for the secured ports. This is done by restoring the TTYC. To restore the TTYC, enter the following command:

```
RST:TTYC=aa: (mml)
```

RST:TTYC= *aa*! **(pds)**
Where *aa* is the TTYC number.

3.5 USING THE LOGIN

3.5.1 GENERAL

When the login process has been successfully completed, the user may continue with a normal session. The only exception may be a message, immediately after login, asking the user to hit the "break" key. This message appears only when logging in as a *UNIX*® RTR operating system-shell system terminal.

3.5.2 LOGGING OFF

When logging off a dial-up recent change terminal or a dial-up *UNIX*® RTR operating system terminal, enter CONTROL-D.

Logging off a dial-up STLWS terminal is more complex because the display and message regions are two separate screens. Perform the following procedure to log off a dial-up STLWS terminal:

(1) Go to the message page (Page 120).

(2) Go to "message mode".

(3) Enter CONTROL-D.

These steps are necessary because the display region is frozen to prevent scrolling off the screen. Since the CONTROL-D is not accepted in "command mode," the user must be in "message mode" to log off. The next user is limited to the bottom of the display if the user is logged on as a *UNIX*® RTR operating system terminal. By going to the message page (Page 120) only, the top few lines are frozen giving a *UNIX*® RTR operating system user more work space.

If a user logs on as a *UNIX*® RTR operating system terminal and is stuck on the bottom of the display, reset the terminal. Reset the terminal by pressing the "reset" key on the terminal or power cycling the terminal.

3.6 PASSWORD PROTECTED COMMANDS

This feature, if enabled by the site, restricts access to the *UNIX*® RTR operating system shell, Office Database Editor (ODBE), and/or Access Editor (ACCED). The following systems may be protected by this feature:

- RCV:MENU:SH
- RCV:MENU:ODBE
- RCV:MENU:ACCED
- RCV:MENU:SCREEN
- POKE 194.

NOTE: Password Protected Commands is ***NOT*** a security feature and can be overridden by most experienced Lucent and telephone company field support personnel. It is provided for a site desiring to block inexperienced personnel and/or clerks from accessing these commands.

The feature is enabled as follows:

From the *UNIX*® RTR operating system, create new logins for each command you wish to protect using the -R option of the **admin** command. The login must be the name of the command followed by a dash (-). Enter the following commands:

admin -a sh- -R	Protects the *UNIX*® RTR operating system shell.
admin -a odbe- -R	Protects ODBE.
admin -a acced- -R	Protects ACCED.

Add a password to each of the following logins as follows:

passwd sh-	Adds a password to the *UNIX*® RTR operating system shell.
passwd odbe-	Adds a password to ODBE.
passwd acced-	Adds a password to ACCED.

After the passwords are defined, the selected command(s) prompts the user for the corresponding password.

From the *UNIX*® RTR operating system, the "Password Protected Commands" feature is disabled by deleting the login corresponding to the command as follows:

admin -d sh- -R	Unprotects the *UNIX*® RTR operating system shell.
admin -d odbe- -R	Unprotects ODBE.
admin -d acced- -R	Unprotects ACCED.

An additional option is also provided that changes the access permissions of the *UNIX*® RTR operating system shell such that the user has read-only permission except in the "tmp" directories. With this option selected, the shell user is not able to run some of the *UNIX*® RTR operating system commands.

To select this option, create the "sh-" login with the '-r' option rather than the '-R' option and enter the following command:

admin -a sh- -r	(Limits the permission shell)

Once this option is selected and the shell is entered, if full permissions to the files are desired, the **su** command and the login 'root' password must be entered.

su root
(root password)

The **su** and **passwd** commands only operate when entered by **RCV:MENU: SH**. These commands do not operate from **RCV:MENU:SCREEN** or **POKE 194**.

3.7 MODIFYING THE *RM* COMMAND DEFAULT OPTIONS

The *UNIX*® RTR operating system *rm* command supports an interactive option ("-i"), which prompts the user for confirmation before removing each file. This provides a second chance to avoid accidentally removing files.

In some situations, it may be desirable to have the *rm* command invoked with the interactive option by default. This can be accomplished by setting up a shell alias for the *rm* command. Perform the following procedure to establish a shell alias:

(1) From the *UNIX*® RTR operating system shell, copy the following provided default shell alias file to the name expected by the *UNIX*® RTR operating system shell. This file contains an alias for the *rm* command.

```
# cp /etc/default.shrc /etc/shrc
```

(2) Exit any active *UNIX*® RTR operating system shells, including those invoked using RCV:MENU or

SCREEN. (The alias file is only read when the shell is invoked.)

The shell alias is now active for all interactive shells, until the preceding *etc/shrc* file is removed. When the *rm* command is used, the user is prompted for each file before it is removed, just as if *rm -i* had been entered. This includes all shells established using **RCV:MENU** and the **SCREEN** program, but does not affect non-interactive shell scripts or applications.

The *rm* alias can be bypassed either by using the full path of the *rm* command (*/bin/rm*) or by executing the *unalias* command. *Unalias* removes the alias for the remainder of the shell session or until the alias is re-established by executing the *alias* command.

See the *UNIX*® RTR operating system shell *sh* manual page for more information.

[redacted]

Figure 3-1 Diagram of Cable from Modem to IOP

```
stlw2xMm;YhM,MOXA:1329:1329:Dialup STLWS:/cft/sh1:/cft/bin/pdshel.app
|_1_||_____2________||_3_||__4_||____5______||__6_____||________7_______|
```

[1] Login id
[2] Encrypted password
[3] User id
[4] Group id
[5] Comment field (optional)
[6] Home directory after successful login
[7] Command to execute after

Figure 3-2 Description of an Entry in /etc/passwd

4. EMACS DESCRIPTION

4.1 INTRODUCTION

EMACS is a screen editor that can be used to create or edit files using a display terminal. The user is shown a display of a portion of the buffer being edited. This display indicates *exactly* what is in the area being displayed, including any non-printing characters. The contents of the buffer being edited can be read from or written to a *UNIX*® RTR operating system file. Characters typed by the user are inserted into the buffer (and reflected in the display) at the point indicated by the terminal's cursor. This is the primary mechanism for entering and modifying text.

Control characters and escape sequences can be used to perform other editing functions, such as moving the cursor to a different position in the buffer, deleting text, replacing text, or searching. Thus, EMACS uses only one mode for interpretating characters, text, or commands that are entered. This eliminates the need to remember what mode is active, and prevents the mistakes that occur when text to be inserted is evaluated as an editor command. A simple mechanism is provided to allow a user to insert control and escape characters when needed.

The most common way EMACS is used to edit is to position the cursor over the area to be changed, and enter the changes. The immediate feedback provided by the visual display appears to be very important to the user.

Table 4-1 provides a summary of EMACS commands.

4.2 BASIC CONCEPTS

4.2.1 GENERAL

Before using the EMACS editing commands, some basic concepts should be learned. EMACS treats the screen and the keyboard differently than the line oriented editors and other screen oriented editors. Some conventions for displaying and inputting characters are different than other *UNIX*® RTR operating system tools, because they were originally developed for another environment.

4.2.2 THE CHARACTER SET

EMACS uses 256 different characters. These include the 128 ASCII characters and 128 "Meta" characters that can be entered from a terminal. A Meta character is entered by preceding it with an escape (**Esc** key).

In this document and in the displays produced by EMACS, control characters are indicated by the character **^** followed by the equivalent printable character (usually capitalized). Thus, **^X** represents a control-x, which is typed by pressing the control and **x** keys simultaneously. For some unusual non-printing characters, the display is not obvious:

^?	Rubout or delete (ASCII 0177)
^@	Null (ASCII 0)
^[	Escape (ASCII 033)
^	The "fs" character (ASCII 034)
^]	The "gs" character (ASCII 035)
^^	The "rs" character (ASCII 036)
^_	The "us" character (ASCII 037).

Meta characters are entered by pressing the escape character and any second character (including a control character). They are displayed by EMACS as **M-** followed by the equivalent ASCII character. For example, **M-a** (Meta - a) is the character obtained by typing escape followed by a, and **M-^B** (Meta - control-b) is the character obtained by typing escape followed by control-b.

4.2.3 THE DISPLAY

The display screen contains a window showing a view of the buffer being edited. A typical display terminal contains about 20 lines. The terminal cursor is positioned at the editor cursor (the position where editing takes place). Each line of the buffer (delimited by a newline character) begins at the beginning of a display line. A line that exceeds the screen width is normally continued on the next screen line. Whenever a line must be continued on the next screen line an exclamation mark (!) is displayed in the last column of the first screen line. If the editor is in line number (*lnumb*) mode, then a line number is printed at the beginning of each line in the buffer.

Printable characters are displayed normally, while tabs are displayed as white space that fills up the space on the screen until the next position at a multiple of eight. Non-printing control characters and meta characters are printed with the preceding conventions. If a file, containing characters that have the high order (parity) bit set, is edited, the characters are displayed as meta characters by EMACS. This only occurs when trying to edit files containing binary information.

In addition to the display buffer, several lines of the screen are used for status information and for displaying parameters entered into EMACS, for example, a filename. One of these lines, known as the status line, contains the editor name, editor version, buffer number and name, and filename. Some of the more recently introduced commands described in this document indicate the version in which they were introduced. This allows a user to determine whether or not a particular command is available in the version running. If the buffer has not been modified since the file was read or written, an **=** is displayed between the buffer and filenames. Otherwise, a **>** appears.

The lines below the status line are used for the time-of-day display (if *time* mode is on) and for EMACS to prompt for parameters for commands. Some commands cause the buffer display to be erased in order to display other information in place of the buffer. When this happens, **Continue?** is displayed at the bottom of the screen. Typing **y**, **' '**, or **Return** brings back the buffer display. Typing **n** may allow the re-execution of the command producing the display.

Figure 4-1 shows a typical screen during a EMACS session. The buffer **Main**, number **0**, is being used to edit a program **test.c**. The buffer has been modified since the last write to the file test.c.

4.2.4 THE TEXT IN THE BUFFER

Each buffer that is edited holds a sequence of characters. Any characters can be present in an EMACS buffer, including control and meta characters. The only limitation is on the number of characters that can be on one line in the buffer. Normally, EMACS treats the buffer just as a sequence of characters.

One difference between EMACS and many editors is that EMACS does not treat "newline" characters specially. Between each pair of adjacent lines of text in the buffer is an invisible "newline" character. If the cursor is at the end of one line, it is in front of the newline character, and deleting a single character deletes the newline, causing the text in the following line to be joined to the current line. Newlines can be inserted, deleted, and searched for like any other characters.

Some EMACS commands operate on units of text in the buffer, like words, lines, sentences, pages, etc. These work on top of the base level which still treats the buffer as a string of characters.

4.2.5 COMMAND STRUCTURE

EMACS does not have distinct "modes" for inserting text into the buffer and for entering commands. Thus, there are no commands for inserting text, and no special convention to end a text insertion. Instead, ordinary characters can be inserted into the buffer at any time by typing the characters, while control and meta characters are used for editing commands.

Each character that is typed into EMACS is in fact interpreted as a command. All of the ordinary printing characters insert themselves into the buffer being edited at the point defined by the cursor. Thus the command invoked when you type the character **x** inserts an x into the buffer at the point shown by the cursor. The control and meta characters are used for various editing functions.

4.2.6 ARGUMENTS AND PARAMETERS

All commands, including the printing characters, take a numeric argument that has some effect on their interpretation. The default argument given to a command for which no argument is specified is 1. To specify some other argument to a command, enter escape followed by a sequence of digits and the command. A negative value for an argument is specified by entering escape followed by **-** and a sequence of digits. Numbers starting with a 0 are interpreted as octal; numbers starting with any other digit are decimal. A second way of specifying the argument is to precede the command by one or more **^U** (control-u) characters. Each **^U** multiplies the value of the argument by 4.

For most commands, the effect of the argument is to multiply the number of times that the command is applied. Thus the sequence **^U^Ux** inserts 16 x's into the buffer at the current location. The sequence **ESC13^N** moves forward 13 lines in the buffer.

In addition to the numeric argument given to all commands, some commands prompt the user for additional character string parameters. For more information about the commands that require parameters and how to enter parameters, refer to Section 4.3.5 ., "Simple File and Buffer Commands."

4.3 BASIC EMACS COMMANDS

4.3.1 GENERAL

Every character typed to EMACS is interpreted as a command. This section describes a simple set of commands that are sufficient for most editing. Subsequent sections describe advanced commands and other aspects of EMACS.

Most EMACS commands have a mnemonic significance that should be obvious (for example, **^B** for backwards or **^D** for delete); others have no obvious meaning. As a general rule, control character commands operate on characters and lines; the corresponding meta character commands operate on words or sentences.

The EMACS user interface was designed for touch typists. Control and meta characters are used for commands rather than the function or "arrow" keys that are different on some terminals. Macros can be used to allow EMACS to respond to the arrow and function keys on a particular terminal, if desired. Typically the only difficulty encountered by an EMACS user in adapting to a new terminal is locating the escape key. With practice, a user adapts to the keys associated with the basic commands.

4.3.2 GETTING HELP OR GETTING OUT OF TROUBLE

The following EMACS commands provide help or correct mistakes.

M-?	Explain □ This command prompts for a character and prints a brief explanation of what that character does.
M-w	Wall Chart □ This command puts a listing of all commands (including user defined commands), and their help explanations into the current buffer. This command is a convenient way of producing a "wall chart" of the commands. The list is inserted into the buffer at the current position; so, one would want to execute it in an empty buffer. Table 4-1 contains a current copy of the wall chart.

^L Refresh □ This command refreshes the display. Occasionally, some error may cause the display to become garbled. **^L** re-creates it from scratch. If **^L** is used with an argument, the number of lines preceding the current line is displayed. When invoked with an argument, this command does not re-create the display from scratch.

^G Abort □ This aborts the current command at any point that EMACS is asking for input. This applies at any step (specifying arguments, typing escape, entering parameters that EMACS asks for, etc.). There are a couple of exceptions related to advanced editing commands, but even with these, typing **^G** several times aborts a command with minimal damage. This is a convenient way to abort any command.

M-u Undo □ This command reverses the last significant text modifying command. Undo is its own inverse; so, invoking **undo** twice, consecutively, reverses the result of the **undo** command. Significant text modifying commands include all commands except the insertion of individual characters.

If an argument is given to **undo**, it is taken as the number of changes to be undone. Each change undone appears as an independent change to the file. This allows the user to "browse" through recent changes. For example, 10 lines are deleted from a file (one at a time) and the **M-5M-u** command is typed, the last 5 lines deleted reappears. If the user decided the last 7 lines are needed, the **M-12M-u** command is entered, undoing the 5 changes made by the first undo and 7 more. The limit on the total number of changes that can be undone depends somewhat on the complexity of the change.

Almost all commands can be undone, but a few effects of commands cannot be undone. **M-u** does not undo the effects of reading or writing files nor can it undo anything done by executing *UNIX*® RTR operating system commands from EMACS. **Undo** cannot be used to bring back buffers that may have been inadvertently killed with the **^X^K** command (see Section 4.3.5). **Undo** can undo all other significant text modifications except when the last significant text modification was a replace command; only a limited number of replacements can be undone. If more replacements were done with one command, **undo** prints a warning error message, and if the user specifies **undo**, it then replaces to the limit.

^X^C Quits EMACS □ This command exits EMACS. If any buffers have been modified since the last write, EMACS asks whether or not to write out each respective buffer before exiting. EMACS does not ask whether to save an empty buffer.

BREAK Interrupt □ Break causes EMACS to stop processing at the next convenient point and displays a message containing a list of options. After pressing the **Break** key, the user can either continue, abort, suspend the present process, or Exit EMACS. EMACS does not respond to the normal interrupt character.

Many commands in EMACS can cause errors to be reported. When an error is reported, EMACS displays a message at the top of the screen followed by a line of underscores, rings the terminal bell, and waits for a response. The user types a space or **y** instructing EMACS to continue attempts to complete the command processing; typing **n** instructs EMACS to ignore the error; typing **^G** instructs EMACS to abort the command causing the error, and **^Z** instructs EMACS to Exit. The responses **^G** and **^Z** are always handled in the same way, while the interpretation of **y** and **n** depends on the error. Generally, the message describes the appropriate dispersement of responses for any error.

4.3.3 SIMPLE CURSOR MOVEMENT COMMANDS

The following commands move the cursor in the buffer without modifying the text in the buffer. Most of these commands use an argument to specify how many times the movement is to be repeated.

^F,**^B** Move forward or backward one character □ The end of each line counts as one

	character; therefore, **^F** at the end of one line places the user at the beginning of the next line.
^N,**^P**	Move to the next or previous line □ EMACS moves to the same character position in the line below (**^N**) or above (**^P**) the current line. Note that if the buffer contains tab or control characters, the same character position may display at different screen positions on different lines.
^A,**^E**	Move to the beginning (**^A**) or end (**^E**) of the current line of the buffer □ These commands work on one line of the buffer not one line of the screen. If the current line is longer than one line of the screen display and continues to the next screen display line, these commands may move up or down on the screen to the real beginning or end of the line in the buffer.
M-<,**M->**	Move the cursor to the beginning (**M-<**) or end (**M->**) of the buffer.
M-f,M-b	Move the cursor forward or backward by one word □ In EMACS, words are delimited by non-alphabetic or non-numeric characters. Backspaces and underscores are considered legal characters in words.
M-a,**M-e**	Move the cursor to the beginning (**M-a**) or end (**M-e**) of the current sentence □ The end of a sentence is defined as a punctuation mark followed by one or more whitespace characters (blank spaces or newlines). With an argument, the respective commands can be used to move forward or backward by a specified number of sentences.
^V,**M-v**	Move to next (**^V**) or previous page (**M-v**) □ The cursor is moved forward or backward so that the display shows the text just before or just after the text that is now displayed on the screen.
M-g	Move the cursor to the line number specified by the argument given to the command.

4.3.4 SIMPLE TEXT DELETING AND MOVING COMMANDS

The following commands delete text from the buffer. These commands operate on text near the current cursor position.

^D,**^H**	Delete forward (**^D**) or backward (**^H**) from the cursor □ The **^H** or **Backspace** deletes the character immediately before the cursor. The **^?** or rubout is a synonym for **^H**. The **^D** deletes the character on top of the cursor. If given arguments, these commands delete blocks of text preceding or following the current cursor position.
M-d,**M-^H**	Delete words forward (**M-d**) or backward (**M-^H**) from the cursor □ These two commands delete words (as defined for **M-f** and **M-b**). If the current cursor position is in the middle of a word, **M-d** deletes from the cursor to the end of the word; **M-^H** deletes everything before the cursor. **M-^?** is a synonym for **M-^H**.
^K	Kill □ Delete to the end of this line. If invoked without an argument, **^K** deletes the remaining text on this line (if any). If no text follows the cursor on the current line, **^K** deletes the newline. With an argument of 0, **^K** deletes the text before the cursor on the current line. With an argument of *n* greater than zero, **^K** deletes *n* lines forward from the cursor position. The text from the cursor up to and including the *nth* line is deleted. With an argument less than zero, the deletion is backwards from the cursor position.
M-	Meta space □ Place an invisible mark on the current cursor position. This mark can be used in subsequent editing. Each mark is simply a position in the buffer (line number and character within the line). Thus, if text added or deleted in front of a position where

a mark was placed, the mark may not remain on the same character but stays on the same position.

EMACS actually maintains 16 different marks, normally allocated as one per buffer. (Thus, if marks are set in different buffers, they are normally independent.) However, this may be altered by specifying a mark number as an argument to Meta-space and other commands that work with marks. This allows the user to mark up to 16 different positions in one buffer. The command **^@** (Control-@) is a synonym for Meta-space, but cannot be typed on all keyboards.

^W	Withdraw □ Delete the text between the current cursor position and the mark. This is a convenient way to delete a well defined block of text. If an argument is specified, it is used to select the mark number. The mark can be either before or after the cursor position and achieve the same effect.
^Y	Yank back □ Insert last killed text. All deleted text is saved in a "kill stack". The kill stack holds the last 16 deletions. There is also a limit on the total amount of text that can be held in the kill stack. The **^Y** retrieves the most recently deleted text. The most frequent use of this command is in moving text around. The procedure is as follows: (1) Kill the text to be moved. (2) Move the cursor to where the killed text is to be placed and enter **^Y**. The **^Y** is also used to undo an unwanted deletion. The **^Y** leaves the mark at the beginning of the inserted text and puts the cursor at the end. The **^Y** treats the argument (if any) as a count for the number of copies of the deleted text to bring back, not as a mark number. The **^Y** operates only with the default mark.
M-y	Replaces last retrieved text □ This command kills the text between the cursor and the mark and replaces the killed text with the next to last item on the kill stack. This command can only be used immediately after **^Y**, where it changes text that has just been retrieved. By entering **^Y** followed by some number of the **M-y**, any text in the kill stack can be retrieved. If an argument is given, it is taken as the number of copies to retrieve.
M-p	Pickup the region of text □ This command picks up the text between the current position and the mark and puts it in the kill stack without changing the buffer. This is useful for duplicating blocks of text in the buffer. An argument can be used to specify which mark to use.
^X^X	(Type control-X twice) □ Exchange the cursor position and the mark. An argument can be specified to indicate the mark with which to exchange.

4.3.5 SIMPLE FILE AND BUFFER COMMANDS

Commands that access files and buffers ask for the name of the appropriate file or buffer at the bottom of the screen. The following commands are used to edit a file or buffer.

^F,**^B**	Move forward or backward one character.
^A,**^E**	Go to beginning or end of line.
^D,**^H**	Delete forward or backward.
^U	Multiply the effect of the next command by 4.

^K	Kill (erase) the whole line.
^G	Abort the command asking for information.
^X	Enter the current line from the file at the cursor.
^Y	Enter the current filename at the cursor.
^L	Redisplay the prompt and the string being entered.
^T	Transpose the characters before and after the cursor.
^Q	Quote the following character (eliminates the special significance of the next character and just sticks it literally in the string being typed).
Return	Ends the line wherever the cursor is located at the time the **Return** key is pressed.

Most file and buffer access commands are invoked through the **^X** command followed by a second character.

In the commands that ask for filenames, the normal shell conventions for partially specified names can be used. Any of the following sequences can be used in a filename and is substituted properly.

$VARIABLE	Substitutes the value of the environment variable **$VARIABLE** or nothing if VARIABLE is not defined. Thus, pathnames like $HOME/.profile can be used.
***** and **?**	Specifies incomplete filenames and are expanded. If more than one file matches the name given, then EMACS picks the first file bearing the name.
~USER	Translates into the home directory of the user USER. In addition, EMACS always translates the special name ~EMACS into an EMACS library directory. This is a directory where special files, needed by EMACS, are stored, and where standard macros are stored (in ~EMACS/macros).
'COMMAND'	Executes COMMAND and substitutes its standard output.

The following commands are used with files and buffers.

^X^R	Read file □ EMACS prompts for a filename, which is entered as described previously. When the filename for **^X^R** has been entered, EMACS reads the specified file into the buffer. Pressing **Return**, in response to the prompt, causes EMACS to read from the current file, if any. EMACS warns the user if the buffer has been modified since the last time the user wrote to it. Many commands in this section, including **^X^R**, use their argument to specify minor variations on the basic action of the command, rather than specifying a count. In the normal case (with the default argument of one), **^X^R** clears the buffer before reading. If **^X^R** is invoked with an argument that is not 1 or -1 (that is, **^U^X^R**) it does not clear out the buffer but instead inserts the file into the buffer at the current cursor position. If **^X^R** is invoked with a negative argument, no error message is produced if the specified file cannot be read. If the file contains a line that is too long for EMACS to handle, an error message warns the user. If the user answers **^G** to the error, EMACS stops reading at that point. If the user answers **n**, EMACS reads the rest of the file, truncating lines that are too long, without further warnings. If the user answers with **y** or with a space, EMACS continues reading and warns the user again if a subsequent line is too long.

^X^W Write file □ Normally, if the specified file exists and has two or more links, EMACS asks whether to overwrite the existing copy of the file or to unlink the specified filename and create a new file in its place, leaving the contents of the old file (which may be obtained through the other names it was linked to) unchanged. If EMACS fails or the *UNIX*® RTR operating system crashes during an attempted write (either **^X^W** or **^X^S**), the previous contents of the file are saved in a filenamed **.EMACS** in your current working directory.

If write access is not held by the user attempting to write to the file, EMACS asks whether a write attempt should be made. If the answer is yes, EMACS tries to write the file by removing the old file and creating a new one. This is not a security hole, but few other *UNIX*® RTR operating system programs write files in this way.

Passing an argument to **^X^W** (that is, **^U^X^W**) causes the contents of the current buffer to be appended to the specified file instead of replacing it.

^X^S Save buffer □ This command writes out the buffer to the last file read or written, if the file has been modified. If the buffer was not read from a file and has never been written to a file, such that there is no filename associated with the buffer, EMACS asks for a filename in which to save the buffer.

^X^B Change buffer □ EMACS allows up to 12 named buffers to be edited concurrently. Each buffer can hold a different file and has its own current cursor position. In EMACS, the user works with one buffer at a time, although two buffers can be displayed on the screen at the same time. The **^X^B** command asks for the name of a buffer and makes that buffer the current buffer.

All of the commands that ask for buffer names accept either the text name of the buffer or the buffer number (shown in parentheses after the editor name on the status line) for a buffer name. The number is convenient if you don't like typing long names. EMACS treats two buffer names specially. For any of the commands that ask for a buffer name, if an empty buffer name is entered, by pressing Return in response to the prompt, EMACS displays all of the currently defined buffers, indicating which buffer is current and which buffers have been modified since they were last written. If a space or **y** is entered in response to the "Continue" prompt that appears after the display, EMACS aborts whatever command asked for the buffer name and continues editing. If **n** is entered in response to the prompt, EMACS asks for the buffer name again then completes the command that asked for the buffer name. If a number is entered in response to the prompt, EMACS completes the command to the buffer with the corresponding number.

If an attempt to create more than 12 buffers is made, exceeding the limit of 12, EMACS indicates this and asks whether to kill an existing buffer that has not been modified. If **y** or a space is entered, EMACS proceeds, reusing an unmodified buffer. If **n** is entered, EMACS suggests other buffers to be re-used until either the user selects a buffer or there are no more buffers suggest. In which case, an error message is displayed and the command attempting to create the buffer aborts.

In all cases, if the buffer name "..." is entered, a new, empty buffer with a unique name is created.

^X^F Find file □ This command prompts for a filename and switches to a buffer that holds the specified file. If the specified file has been read into a buffer, the effect of "find file" is to change to that buffer. If no buffer holds the specified file, the effect of "find file" is to read the specified file into a newly created buffer. "Find file" is a convenient way to switch between several files while editing. If **^X^F** is invoked with a negative argument, no error message is produced if the specified file cannot be found. If given an argument **>1**, "find file" automatically reads in a fresh copy of the file. If "find file" is given a negative argument and the specified file does not exist, a message is not given, but an empty buffer with the given name is created.

^X^K Kill Buffer □ This command prompts for a buffer name and destroys the specified buffer. Text in the buffer that is killed is lost and cannot be recovered. A current buffer cannot be killed using this command.

4.3.6 SIMPLE SEARCH AND REPLACE

EMACS provides commands that search for text in the buffer and allow global replacements. (For example, change every instance of "football" to "baseball".) The following paragraphs contain descriptions of the simple "search and replace" commands. Also listed are useful miscellaneous commands. For information about complex "search and replace" commands, refer to Section 4.4.4 ., "Advanced Search and Replace Commands."

^S,^R Forward and Reverse Search □ These commands are used to search for text in a buffer. EMACS displays, at the bottom of the screen, a "Search" or "Reverse Search" prompt. At the prompt, the desired text or character (search string) is entered, and EMACS begins to search, from the current position in the buffer, forwards **^S** or backwards **^R**, for the first occurrence of the search string. Once EMACS finds the search string, the data is displayed with the cursor positioned at the beginning of the located data. In addition to typing normal characters as search strings, the following special characters may be used to either edit the search string or control the search.

^H Deletes the last character of the search string and causes the cursor to locate and display the first occurrence of the modified search string.

Esc Aborts the search, leaving the cursor at its last location.

^G Aborts the search and returns to the original position in the buffer.

Return or **Newline**
Causes a newline to become part of the search string. The Newline is displayed as **^J** (which is the control character actually used by *UNIX*® RTR operating system to indicate "Newline") in the search string. Thus, if the sequence **^Sthe~end<<return>of** is the search string, EMACS searches for "the~end" at the end of a line and "of" at the beginning of the next line.

^S or **^R** Changes the direction of the search. After a **^S** (forward) or **^R** (backward) is entered, the *last* successfully found search string becomes the current search string.

^Q Quotes the next character, making it part of the search string, ignoring its usual significance. This is a way to search for strings that contain control characters.

Other Control Characters
When entering "Other Control Characters", EMACS searches, finds, and executes the control character successfully found.

In EMACS, this search is called an incremental search because it shows what is matched incrementally as it is entered. For an incremental search, the search string and the desired string must match *exactly*. (A more complicated search that allows pattern matching is described in Section 4.4.4 .) Incremental search stops when the beginning or the end of the buffer is reached.

Normally, search considers uppercase and lowercase letters to be different; however, the user can override this with *caseless* mode.

M-r Query replace □ A *From* string and a *To* string prompts the user. Each can be edited using the conventions described in the previous section for editing filenames. In general, "Query replace" allows the user to replace all of the strings in the buffer from the current cursor position to the end of the buffer that match the *From* string with the *To* string. In the *To* string, the **&** character can be used to designate replacement with the *From* string. To get a real **&**, prefix it with a **'**´. To get a real **'**´, prefix it with another **'**´. If the single character **%** is specified as the *From* string, the last string searched for (either from **^S** or **M-r**) is used. Note that a newline can be specified in the *To* string either by entering it explicitly as **^Q^J**, or by putting the string **\n** in the string where the newline is to appear. EMACS then searches for the *From* string, positions the cursor in front of it, and prompts the user. The following commands are used to control the replacement of an item:

<space> or **y**	Replace this occurrence and move to the next occurrence.
n or **^?**	Skip this occurrence and move to the next occurrence.
.	Replace this occurrence and exit "query replace".
<	Go back to the last replacement and stop. (Useful for correcting mistakes!)
^G	Quit □ Exit "query replace" without replacing the current match.
b	Go back to the previous occurrence of the *To* string. It does not find an occurrence that the user has already replaced.
r	Replace the rest without question, and show the result of the replacement after each.
R	Replace the rest silently. (That is, do not show the result after each replacement.)
<Esc>	Causes EMACS to ask for a new string to replace the *To* string. The current occurrence is replaced with the new *To* string, and EMACS goes to the next occurrence.

Normally, "query replace" shows all occurrences of the search string. With an argument greater than 1 (that is, **^UM-r**), it looks only at the first match of the *From* string on each line. "Query replace" exits when the *From* string is no longer matched. **?** prints a summary of the options. As with incremental search, the *From* string must match exactly to the pattern in the buffer. There is a more advanced form of "query replace" that allows pattern matching and is described in Section 4.4.4 .

4.4 ADVANCED EDITING COMMANDS

4.4.1 GENERAL

The advanced editing commands are available for the following situations.

□ Inserting odd characters

□ Commands related to windows

□ Advanced search and replace commands

- Macros, keyboard macros, and input files
- Commands that escape to the *UNIX*® RTR operating system
- Miscellaneous commands.

4.4.2 INSERTING ODD CHARACTERS

Because EMACS uses control and escape characters for commands, these characters cannot be typed directly into the buffer. The following commands are used to insert control and escape characters into a buffer.

^Q	Quote the next character(s) □ The **^Q** accepts one or more characters (the number of characters specified by its argument) from the terminal and inserts them "blindly" into the buffer without interpretation. Only the newline (line feed) character is interpreted. EMACS strips the parity bit from all characters read from the terminal; so, all characters inserted this way have zero parity.
M-q	Quote characters and turn on parity bit □ This acts like **^Q**; however, it turns on the parity bit in the character before inserting the character. Characters inserted this way are displayed as meta characters by EMACS.
M-	Convert the argument to a character and insert into the buffer □ This command takes an argument, converts the argument to a character, and then inserts the character into the buffer. This is occasionally useful for inserting odd characters for which the ASCII code is known.

4.4.3 COMMANDS RELATED TO WINDOWS

EMACS provides the ability to display two buffers on the screen at the same time. When this is done, the screen is split vertically, and one buffer is displayed in the top half and one in the bottom half. The status line shows the status of the current buffer.

When EMACS displays two buffers, only the current buffer is actively updated. The second buffer remains inactive until the user is active in the second display screen. The same buffer can be displayed in both windows. However, the current position is associated with a buffer, not a window; thus, if the current position is in the lower window, when the user moves to the upper window, the cursor immediately moves to the same position in the upper window where it resided in the lower window. If you have different buffers in the two windows, the current position in one buffer is independent to the buffer in the other window.

^X2	Enter two-window mode □ EMACS asks for a buffer to show in the second window. The current buffer becomes the upper window; the requested buffer becomes the lower window and the current buffer.
^X^^	Grow window □ The current window grows by the number of lines specified in the argument **^X^^**. A negative argument can be used to shrink the current window. With only one window on the screen, this command can also be used to grow or shrink the buffer display. This can be useful in avoiding long delays when working from a low speed terminal port.
^X1	Return to one window □ The current window grows to fill the screen.
^X^O	Switch windows □ Make the dormant window current and the current window dormant.

4.4.4 ADVANCED SEARCH AND REPLACE COMMANDS

The following commands allow pattern matching of regular expressions, comparable to those used by the ed editor.

For a brief description of regular expressions, refer to the manual pages for *ed(1)*. EMACS provides some additional special sequences for regular expressions. The character sequences **\<** and **\>** can be used to match the beginning and end of words. Thus, the string "\<the\>" matches any occurrence of the word "the," but not any word containing the sequence of letters "the," such as "other." EMACS also allows the user to specify a newline as one of the characters to be matched anywhere in the expression. Specify a newline as **\n** either by itself or as an alternative in a set of characters (that is, **[a-z\n]**). Neither the special symbol **.** nor a negated set of characters (that is, **[^a-z]**) match a newline to avoid unexpected results of expressions such as "**.***".

In addition, a set of alternatives can be specified by expressions of the form: \(eps1\|exp2\|exp3\). This expression matches the text matched by exp1, exp2, or exp3, and saves it for later reference as with other expressions enclosed in parentheses. Any number of alternatives may be specified. At present, *****, **+**, and **\{n,m\}** cannot be used after a set of alternatives to specify multiple matches. (Note that the user can specify multiple copies of text matching one of the alternatives with an expression of the form: \(exp1\|exp2\|exp3\)\1*.

In constructing regular expressions, it is important to prefix a special character (for example, **.** or *****) with a backslash ****. To quote control characters in regular expressions for EMACs, type **^Q** before the control character.

Constructing regular expressions that match efficiently can be difficult. Expressions beginning with a normal text character are generally matched much more efficiently than others. An expression such as [a-c] is matched much more efficiently than the equivalent form \(a\|b\|c\). Expressions matching an indefinite number of alternatives by using ***** or **+** are probably the slowest to execute, particularly when the expression is likely to match a large number of characters and is not the last expression being matched (that is, **.*x**). Be especially careful of expressions that match an indefinite number of characters including newlines (for example, **[a-z \n]***), as these can trigger a very long search that extends over many lines.

M-^S Regular Expression Search This command prompts for a regular expression for which to search. An expression may be edited the same as editing filenames. Pressing **Return** in response to the prompt repeats the last search made using the **M-^S**, **^S**, or **^R** command.

Depending on the argument given to the search command, a forward or backward search can be made ending at the beginning or end of a buffer, or wrapping around (like ed).

1	(default) Search forward, wrapping from the end of the buffer to the beginning, and failing only if the buffer contains no match for the given string.
-1	Search backwards, wrapping around from the beginning to the end of the buffer.
> 1	Search forward, stopping at the end of the buffer.
< -1	Search backward, stopping at the beginning of the buffer.
0	Search the current line only. The search finds the indicated expression anywhere on the current line and fails otherwise.

In all cases, EMACS can repeat the search, looking for the next (or previous) occurrence of the search string by typing **^S** or **^R** immediately after the regular expression search.

M-^R Regular Expression Query Replace □ This is the same as "query replace," except a regular expression is allowed in the *From* string. The special character sequence, \<digit>, may be used in the *To* string to specify that the characters matched by the **n**th subexpression (delimited by **nd \)**) are to be used in the replacement string.

4.4.5 MACROS, KEYBOARD MACROS, AND INPUT FILES

EMACS provides many methods for a user to construct editor programs from sequences of commands. The macro programming facility is the best way to construct substantial programs. Among the following commands are two other methods for saving a sequence of EMACS commands for later use, input files and keyboard macros, and the commands that load and invoke "full" macros to use in EMACS.

^X^I Redirect input □ This command directs EMACS to take input from a file. The file is assumed to contain EMACS commands and can be created by editing with EMACS, using **^Q** to enter control and escape characters. An input file can also be created by using the commands to create keyboard macros described in the following paragraphs, and then saving the resulting keyboard macro file.

This command can be used to perform a series of commands on the current buffer or to set up a standard set of initializations. Thus, the file is to contain *exactly* the same data as a command typed from a keyboard to perform whatever task. If a file contains only printable ASCII text, tabs, and newlines, **^X^I** effectively reads the file into the buffer at the current location. However, this is very slow; therefore, the **^X^R** command is preferred.

A file suitable for executing with **^X^I** is known as a keyboard macro, because it is interpreted just as if it had been typed from the keyboard. The following commands provide a sensible method to create and execute keyboard macros.

^X(Begin Keyboard Macro □ This command starts remembering the keystrokes entered. These keystrokes can later be executed as a keyboard macro.

^X) End Keyboard Macro □ This command stops remembering keystrokes for a keyboard macro.

^XE Execute Keyboard Macro □ This command retrieves and executes the keystrokes typed between **^X(** and **^X)**. EMACS executes these macros the same as commands entered from a terminal. Keyboard macros are saved in the file .emacs_kbd in the home directory. These are saved between sessions, so that **^XE** is the same as invoking **^X^I** (Input file) and giving *$HOME/.emacs_kbd* as the filename to execute. Remember, **^X(** and **^X)** can be used to create a keyboard macro and saved for later use by moving the *$HOME/.emacs_kbd* file to another file. The saved macro may be invoked using the **^X^I** command with the name of the saved file.

^Xd Define macros □ This command treats the current buffer as definitions of new macro commands. The commands are defined and become available for use. For a complete description, consult the macro programming manual. The **^Xd** command should not be used with the file created from a keyboard macro.

^X^L Load macros □ This command allows the user to load "full" macro definitions from a file. It is described in the macro programming manual; however, even if the user does not program macros, the user may use macros defined by others by using the **^X^L** command to load the resulting files. On some systems on which EMACS is installed, there is a library directory of macros available for general use in *~EMACS/macros*, and the file *~EMACS/macros/CATALOG* gives a catalog to the available macros.

M-x Execute Macro command This command asks for a macro name and tries to execute it. If there is no macro currently loaded with the name given and if *autoload* mode is on, EMACS tries to load the file with the name of the macro from the directory $EMACS_LIB (if this environment variable is defined) or in the directory ~EMACS/macros if it hasn't been found yet. If the macro cannot be found, an error results.

^Z Exit level In an EMACS editing session, a nested level of EMACS can occur by typing **break** and responding **y** to suspend whatever task is being performed and invoke a new command interpreter, or by invoking a macro that uses the recursive edit command to allow editing something from inside of a macro. The **^Z** exits your current level of EMACS, returning to whatever called it.

The **^Z** command performs a function similar to **^X^C**. Ordinarily, both commands exit from EMACS; however, if the **Break** key or **^Z** key is pressed while executing a macro that allows editing the buffer and returning to the macro, **^Z** returns to the break or the macro instead of exiting.

4.4.6 COMMANDS THAT ESCAPE TO THE *UNIX*® RTR OPERATING SYSTEM

The following commands interact with the *UNIX*® RTR operating system, allowing the *UNIX*® RTR operating system commands to run or send mail from inside EMACS.

M-!,**M-$** *UNIX*® RTR operating system escape.
These two commands implement five different methods to run *UNIX*® RTR operating system commands from EMACS. In all cases, the commands prompt for the name of a *UNIX*® RTR operating system command to run then executes the command. The command runs through the normal shell, as indicated by $SHELL. If the special command name **sh** is entered, the normal shell is run instead of **sh**. To facilitate writing programs that interact with EMACS, the environment variable "filename" is set to the name of the current file in EMACS when the command is run. The following commands are available:

M-! Run the command, suspending EMACS while the command runs.

^UM-! Run the command and feed it the contents of the current buffer as standard input. When the buffer is exhausted, the command sees an "end of file."

M-$ Run the command with standard input from the terminal. Standard output and standard error are captured in the buffer ".exec," replacing the current contents. If the buffer ".exec" does not exist, EMACS creates ".exec." Normally, output from the command is also displayed on the terminal as it is produced, but this can be overridden via the *usilent* or *noecho* modes described in Section 4.5.3 . (For example, saving a copy of the error messages produced by a C compilation of a file being edited.) The filename of the ".exec" buffer is set to the command line that produced it. This can be useful to re-execute the same command. Make ".exec" the current buffer; enter **M-$** and **^Y** followed by newline as the command line**^Y** gets the old command line back, and newline executes the command.

^UM-$ Run the command and append the output to the buffer ".exec". This is the same as the previous command description except the current contents of

	the ".exec" buffer is not replaced but appended by the output of the command. EMACS also inserts the command line into the .exec buffer when the command is run.
^X^D	This command directs EMACS to change the current working directory. If buffers with filenames that are relative pathnames (not starting with '/') exist, change the working directory and then save one of the buffers; the buffer saves in a different file because of the change in the working directory.
M-^M	This command identifies the current buffer as *UNIX*® RTR operating system mail and mails it. The buffer must contain at least one line starting "To:" , which specifies the recipients of the mail. Each recipient is delimited by a space. Any number of recipients may be listed in a single line; however, to improve readability, additional "To:" or "Cc:" lines may be used in specifying lists of recipients. Any errors encountered by mail are printed. If the environment variable *$MAILER* is set, it is taken as the name of the command to run to send the mail; otherwise, EMACS runs "mail."

4.4.7 MISCELLANEOUS COMMANDS

The following miscellaneous commands are available.

^O	Create an empty line □ This command creates one or more empty lines at the current cursor position. This is useful for inserting text in the middle of the buffer, while minimizing the amount of screen refresh needed.
^T	Transpose the next two characters □ This command moves the cursor forward one character for each transposition, such that giving **^T** a count as an argument causes the character at the cursor to be dragged forward through the text.
^X^T	Transmit text to another buffer □ This command sends the text between the mark and the current cursor position in the current buffer to another buffer. EMACS prompts for the name of the other buffer, and the text is inserted into that buffer at the current cursor position for that buffer. The current buffer remains unchanged. If an argument is given, EMACS selects the mark to use. If the target buffer has a sub-process running under it, the region is also sent to that process and is always put at the end of the buffer. (See the description of **M-$** in Section 4.4.6 for more information.)
M-s	Display statistics □ This command displays statistics about the editing session, such as how many characters EMACS has sent to the user and how many characters the user has typed.
^X=	Display status □ This command displays status information (current line, number of lines in buffer, current character position, number of characters in buffer, etc.).
M-/	Begin comment □ This command begins a C program comment by moving to the appropriate column (specified by *comcol* mode) and putting a **/*** in the buffer. (If the cursor is in the first column, the comment is not indented.) The next newline closes the comment and automatically appends a ***/**. If the fill mode is on and the comment line is wrapped onto the next line as a result, the next line begins with * , and the comment stays open until closed with a newline.
M-_	Underline word □ This command underlines the following word of text using backspaces and underscores. This can be used to generate underlined text that is displayed properly by most printers and formatting software.
^C	Capitalize □ This command capitalizes the letter under the cursor and moves the cursor

forward one position. Lowercase alphabetic characters are converted to uppercase while other letters are unchanged.

M-c Capitalize word □ This command capitalizes the letter under the cursor, and the cursor is moved to the beginning of the next word.

M-l Lowercase letter □ This command converts the letter under the cursor to lowercase and the cursor is moved to the right.

M-~ Unmodify Buffer □ This command causes a buffer to be marked as being unmodified even if the buffer has been modified since the last write. Doing this avoids having EMACS ask whether or not to write the buffer when the user exits EMACS and knows a write is not necessary. With an argument greater than 1 (that is, **^UM-~**), this command marks the buffer as modified.

M-^L Redisplay top □ This command redisplays the window with the current line at the top. This is useful for viewing the lines that follow the current line. Note that this does not re-create the entire display, as does **^L**; so, a garbled screen is not necessarily cleared.

M-" Auto Fill Buffer □ This command re-adjusts the lines in the buffer so that each line contains 72 or fewer characters. The adjustment is done by moving words from one line to another. The nroff or mm command lines and blank lines are preserved as is. The adjustments made generally do not change how the file would look after being run through nroff/troff unless the text contains tables, displays, or other fixed format text. Lines are broken at white-space (space or tab) characters.

If you give **M-"** an argument other than 1, only the part of the buffer between the cursor and mark is readjusted.

M-: Remap character □ This command remaps character commands by prompting for a character sequence (a single character or a meta or **^X** sequence) and a command (also specified by a character sequence) to put on that character sequence. This allows the user to reconfigure EMACS as needed. Any character, including characters like **^U**, **^X**, or escape; however, to re-map escape or **^X** or to map other characters to these "commands", **M-:** must be invoked with an argument of -1 (**M--M-:**). With a negative argument, **M-:** asks for only single characters for the sequences to map. The command sequence is always interpreted with the default bindings (as documented in this memo), and not with the bindings set up with earlier **M-:** commands. Thus, **M-:^A^B** followed by **M-:^B^A** transposes the **^A** and **^B** commands, since the command **^A** in the second **M-:** command refers to the default meaning of **^A**, not the meaning established by the first **M-:** command. **M-:** also changes the behavior of the control characters used to edit filenames and other string parameters but does not change the behavior of some of the characters that have special meaning in response to prompts issued by various commands, such as the responses to "query replace."

With an argument of 4 (**^UM-:**), this command asks for a character and a macro name to assign to that character. This allows binding of macro commands to characters by their names instead of their current character binding.

With an argument of 0 (M-0M-:), this command resets all of the keyboard character bindings to their default values. This may be useful in recovering from trouble.

With an argument of -1 (**M--M-:**), the character sequence and command are single characters, allowing escape and ^X to be remapped to other commands, and allowing other single characters to assume their function. Note that **^X** and escape "commands" can only be attached to single characters.

In all cases, the bindings established with this command and through defining macros

	apply to both character commands typed from the keyboard and to characters in keyboard macros and initialization files. The bindings established with this command do not apply to the character commands executed in the body of "full" macros.
^X^M	Set Mode □ This command can be used to set parameters to customize the behavior of EMACS. For more information on modes, see Section 4.5 .

4.5 MODES

4.5.1 GENERAL

EMACS has a variety of parameters that can be changed from commands entered in the terminal. These are referred to as "modes" in this document and in the messages printed by EMACS. There are two types of modes: on/off modes and integer modes. The **^X^M** command can be used to display or set these parameters. The **^X^M** command prompts for the name of the mode to set. If a return is entered in response to the prompt, the current mode settings are displayed. Normally, EMACS displays the value of each integer mode and the name of each on/off mode that is currently on. If **Return** is entered in response to **^U^X^M**, EMACS also displays the names of the on/off that are currently off, indicating for each mode whether the mode is on or off.

Modes are set by giving the name of the mode to set in response to **^X^M**. If an on/off mode is given, it is turned on if no argument is given to **^X^M**, and turns it off if an argument other than 1 is specified. (Thus **^X^M** turns on, **^U^X^M** turns off). For an integer mode, the mode is set to the value of the argument.

The modes and their types are listed in the following sections, along with their default values. For ON/OFF modes, the default is **highlighted**. The modes are grouped into four broad categories:

- □ Display modes change how the information in the buffer is displayed but have no effect on its content.
- □ Interface modes change the command interface to EMACS in minor ways but do not really change any of the behavior of the commands.
- □ Command modes change the way in which some of the commands work.
- □ Terminal modes change the way EMACS uses the terminal, and exist mainly to get around special problems caused by certain kinds of terminals.

4.5.2 DISPLAY MODES (PARAMETERS)

lnumb	Line Number Mode (**ON**/OFF) □ This parameter causes the current line number to display at the left of each line.
lnowid	Line number width (INTEGER=4) □ This parameter specifies how many character positions are reserved for the line number when in line-number mode.
height	Display height (INTEGER=<screen_size-4>) □ This parameter dictates how many lines from the buffer are displayed on the screen. The number of lines is automatically set based on the terminal type whenever the terminal type is set, and is changed by the one-window and two-window commands. This mode and *width* mode can be set explicitly to restrict the display to a subset of the entire terminal screen or can be used to allow the user to use a terminal with a settable screen size for which EMACS does not have the right size built in.
width	Screen Width (INTEGER = <set based on terminal type>) □ This parameter specifies the width of the display screen.
tabstop	Tabstop interval (INTEGER=8) □ This parameter is the number of characters per tab that

	are displayed. A deeply indented C program may be more readable if *tabstop* is set to something smaller than the default value of 8.
backspace	Display of backspaces (ON/**OFF**) □ Turning on *backspace* parameter causes backspace characters (**^H**) to display as moving back one column rather than as a **^H**. This is very useful for viewing nroff output or manual pages, but editing the resulting text can be a bit tricky, because it is impossible to tell whether the character under the cursor is the one being displayed, one that has been overprinted, or a backspace. *Backspace* mode is set on automatically if your terminal can display underlined text.
time	Display time (ON/**OFF**) □ When *time* parameter is on, EMACS displays the time of day below the mode line (the one that says EMACS and the buffer name). The time is updated every time a character is read. Using *time* mode when entering lots of text is expensive in processor cycles.
display_percent	Display cursor as percent of buffer (ON/**OFF**) □ When *display_percent* parameter is on, EMACS displays the percentage of the current buffer beyond the current cursor position on the mode line.
7bit_ascii	Display only 7-bit characters (ON/**OFF**) □ This parameter controls how characters with the high-order bit set are displayed. With this mode OFF, the high-order bit characters are displayed as **M-** followed by the character. With this mode ON, the high-order bit characters are displayed as highlighted (underlined) characters. This mode is most useful for editing files used with personal computer word processing systems which use the high-order bit for formatting control.
leftmargin	Leftmost displayed column (INT=0) □ In picture mode (See Section 4.5.4), EMACS automatically scrolls the window left or right to keep the cursor on the screen. This mode allows altering this behavior. The value of *leftmargin* parameter is the leftmost displayed column. Setting the left margin causes the screen to scroll left or right. In all cases, if the cursor wanders out of the window, EMACS picks a *leftmargin* parameter to keep the cursor on the screen.

4.5.3 INTERFACE MODES

save	Automatic buffer saving (ON/**OFF**) □ If *save* mode is on, EMACS automatically writes the current buffer after *savetype* characters have been entered since the last save. EMACS also automatically saves the current buffer before any command that changes buffers or runs a *UNIX*$^{\circledR}$ RTR operating system command, such as **^X^B**, **^X^F**, **^X^O**, **^X^D**, **M-!**, and **M-$**. This mode reduces the chance of disaster in the event of a crash but may delay editing by causing a lot of extra writing to the file.
savetype	Save type ahead (INTEGER=256) □ If *save* parameter is on, this is the number of keystrokes between saves.
verbose	Verbose prompting (**ON**/OFF) □ When *verbose* mode is on, EMACS prompts for more input when **^X**, **^Q**, **Escape**, or **^U** are entered. This eases the tracking of the users location. *Verbose* mode also effects some error messages. In general, turning the *verbose* mode off causes some error conditions (such as using one of the commands that uses the mark when the mark has not been set) to be handled by supplying a default answer, rather than printing an error message.
keepscroll	Lines kept when paging (INTEGER=0) □ This parameter specifies how many lines are to be preserved on the screen when forward page or backward page is invoked.
smoothscroll	Smoothscrolling (**ON**/OFF) □ When this parameter is on, EMACS tries to scroll text onto

	the screen smoothly, one line at a time when moving forward or backward by less than one screen full or when inserting or deleting lines. With this mode off, any insertion, deletion, or screen movement is done all at once. The display is somewhat slower with smoothscroll on but may be easier to read when scrolling forward through text.
caseless	Ignore case in searches (ON/**OFF**) □ This parameter causes either case characters in the search string to match either case in the buffer on all searches and query replace.
mailtype	Check mail interval (INT=100) □ This parameter determines the number of input characters between checks of your mailbox (**$MAIL**). When EMACS discovers something in the mailbox, a warning is displayed at the bottom of the screen.
end_newline	Behavior of ^N at end of buffer (ON/**OFF**) □ This parameter determines whether executing the **^N** command, in the last line of the buffer, adds a new line to the buffer (mode ON) or whether it signals an error (mode OFF).
usilent	Silent *UNIX*® RTR operating system commands (ON/**OFF**) □ If this parameter is on, EMACS does not display the command name or output for **M-$**. This is useful for invoking a *UNIX*® RTR operating system command in a macro and capturing the output without disturbing the display.
noecho	Don't echo **M-$** output (ON/**OFF**) □ If the parameter is on, EMACS does not echo the output of commands run with **M-$** to the terminal. The difference between *noecho* and *usilent* is that *usilent* isolates commands run from EMACS from the terminal completely, so that EMACS does not clear the screen and does not have to redraw the screen when the command exits. *noecho* has no effect at all on commands run via **M-!** but does eliminate the display of output from **M-$**. It is most useful when running something that produces lots of output, capturing the output in ".exec".
eofnl	Write newline at end of file. (**ON**/OFF) □ When this parameter is on, EMACS appends a newline character to any file written by EMACS from a buffer that does not end in a newline. EMACS allows the creation of files that do not end in newline. Unfortunately, many *UNIX*® RTR operating system tools get confused when reading such a file. This mode is on by default and prevents writing a file that causes troubles. Turn this mode off when editing files that are not to end in a newline.
savelink	Preserve links on write (ON/**OFF**) □ When this parameter is on, EMACS to automatically write into the existing file when writing to a file with multiple links, instead of asking the user what to do in this situation.
search_newline	Newline ends search (ON/**OFF**) □ When this parameter is on, EMACS causes incremental search to stop if a newline or carriage return is typed from the terminal.
autoload	Automatic macro loading (**ON**/OFF) □ This mode controls the automatic loading of macro packages when an undefined macro is called. If *autoload* mode is on and an undefined macro by the name of <name> is called, EMACS first attempts to load $EMACS_LIB/<name> (resolving the environment variable $EMACS_LIB), and then tries to load ~EMACS/macros/<name>. If either of these files exists and defines a macro called <name>, that macro is called and execution continues; otherwise, an error message is produced. This allows macros to be loaded incrementally, only when needed. With the mode off, any attempt to call an undefined macro results in an error.

4.5.4 COMMAND MODES

fill	Automatic line filling (**ON**/OFF) □ When this parameter is on, EMACS automatically moves to the next line whenever the cursor moves to the end of the line, breaking the line at a

word boundary (programmable by changing the character tables). This is very useful for entering text, as no newlines need be entered in the middle of the text.

fillcol Right margin for *fill* mode (INTEGER=72) □ This parameter is the character position beyond which *fill* mode causes the line to be broken.

c C source indentation (INTEGER=0) □ This parameter controls automatic indentation. When set to 0 or 4, indentation is off and each new line is started at the left-hand edge. When this mode is set to 1, each line is indented with the same number of tabs as the one before it, adjusted for the number of opening and closing braces in the previous line. If the file being edited also has a name that ends in ".h", or ".c" (conventionally indicating a C program), the indentation of each line is adjusted for labels and case declarations (which are indented one level less) and preprocessor statements (which always start at the beginning of a line). The indentation of a line may be adjusted whenever any of the characters **:**, **#**, or **}** are typed to make it conform to the normal rules for C program indentation, allowing C source to be entered without adjusting indentation. The special behaviors for C source can be forced to occur in other files (not ending in .c or .h) by setting C mode to 2.

The C mode is useful for both text, where it can be used to maintain indentation in indented paragraphs, and source, where it allows you to enter text without indenting it and usually get a properly indented result.

comcol Comment Column (INTEGER=40) □ This parameter is the column in which comments entered by using **M-/** begin.

rigid_newline Rigidly insert newlines (/**OFF**) □ This parameter causes any newline to insert a newline into the file. With this mode off (the default), a newline does not insert anything if the following line is empty, but simply moves to the next line.

readonly Read-only buffer (ON/**OFF**) □ When this parameter is on, saving the current buffer is disabled. The **^X^S** complains with an error message; autosaving does not take place, and EMACS does not complain if an exit is attempted without writing buffers.

picture Tailor editing for two dimensional displays (ON/**OFF**) □ When this parameter is on, EMACS treats the buffer as an "electronic blackboard" rather than the exact contents of a *UNIX*® RTR operating system file. The display shows a rectangular region of the blackboard through a window. Characters beyond the right margin are not displayed but are indicated by a '!' in the right margin. The window is moved left or right to keep the cursor in the window. The *leftmargin* mode gives some explicit control over the window, but, in general, it is self adjusting. If the window is not at the left edge of the blackboard, then the character offset of the left edge of the display is given on the mode line, in front of the editor name.

Picture mode changes the behavior of several basic commands to be more suitable for two-dimensional editing.

^N/^P These move to the same column in the target line, extending the target line with spaces if necessary.

^F/^B These do not move off the current line. The **^B** sticks at the left margin; **^F** continues to extend the line to the right if necessary.

^W/^Y These treat the region to be deleted as a rectangle, bounded by the mark and the cursor position at the corners. All text in the rectangle defined by these positions is killed by **^W**. The **^Y** brings text back in the same way. All text deleting commands behave like **^W**, in that the start and end positions of the text region to be deleted are taken as corners of a rectangle. When the text region being deleted is all on one line, behavior is identical to normal EMACS; however, deletion across line boundaries by using commands like **M-d** may not do anything sensible.

Picture mode is most useful with *nodelete* mode, *notabs* mode, and *overwrite* mode all set. Picture mode can be used without the others set, but the presence of tabs or control characters in the buffer may give unexpected results when navigating in the file. This mode is particularly useful for editing fixed format tables and picture images of "typewriter art."

overwrite Overwrite instead of insert (ON/**OFF**) □ When the *overwrite* parameter is on, text entered overwrites existing text. Text entered when the cursor is at the end of a line is inserted as before. *Overwrite* mode may be more natural for some people when making corrections.

nodelete Don't close deletions (ON/**OFF**) □ When this parameter is on, text deleted in the file is not closed up but instead is overwritten with spaces. After any deletion, the cursor is moved to the first character of the region deleted. The *nodelete* mode should probably be part of the *overwrite* mode; however, the*overwrite* mode considerably pre-dated the *nodelete* mode and was left alone for upward compatibility.

notabs Eliminate tabs (ON/**OFF**) □ With this parameter is on, EMACS does not display tab characters in the buffer as white space but instead shows them as **^I** (control-i), which is the ASCII code for a tab. When entering a tab by typing **^I** or tab, EMACS converts it to the proper number of spaces to come up to the next tabstop column. EMACS does not convert tabs already in the file, though there are macros that do this.

The main use of this mode is in creating and editing fixed format information in *picture* mode, though it can also be useful in showing where tabs are in the buffer.

4.5.5 TERMINAL MODES

nobell No Bell (ON/**OFF**) □ Ordinarily, unexpected conditions, such as errors or quitting out of commands, cause the terminal bell to ring. Turning on the *nobell* parameter prevents EMACS from ringing the terminal bell.

tspeed Terminal Speed (INT=<terminal dependent>) □ This parameter is the speed of the terminal in milliseconds per character. This parameter, set whenever EMACS is entered, is used in determining how to update the display most efficiently.

controlify Controlify mode (ON/**OFF**) □ When this parameter is on, EMACS causes a particular character (normally, **^^**, but settable to another character with the ctl_char described later in this paragraph) to act as a prefix specifying that the next character is to be interpreted as a control character. This mapping takes place at any time, not just when entering commands. Thus, the sequence **^X^^s** is mapped into **^X^S**, and causes the save command to be invoked. The sequence **^^q^^m**, typed in response to a request for a filename, is mapped into **^Q^M**, which is interpreted as asking for a filename of **^M**. The *controlify* parameter is primarily intended to allow characters to be entered which cannot be sent transparently from the terminal to EMACS. The most frequent examples are **^S** and **^Q**, which cannot be sent by some terminals and some local area networks. By setting the *controlify* parameter, these characters can be sent with **^^s** and **^^q**.

ctl_char Prefix character for *controlify* (INT=30) □ This parameter sets the value of the character used when in the *controlify* parameter to indicate that the next keystroke should be interpreted as a control character. It is the ASCII value of the character to be used.

flow_lim Flow Control Limit (INT=0) □ This parameter enables the use of xon/xoff flow control during output to control the rate of output to the terminal. EMACS normally turns xon/xoff flow control off to allow **^S** and **^Q** to be passed to EMACS to be used in specifying commands. If *flow_lim* is non-zero, then whenever more than *flow_lim* characters are sent to the terminal at once, xon/xoff flow control is temporarily enabled to allow the terminal to send **^S** to request that the *UNIX*® RTR operating system stop output. The xon/xoff flow control is disabled when output is complete. The only effect that is noticed is that input is terminated while EMACS is updating the screen. Normally, EMACS supplies sufficient padding to allow running without xon/xoff flow control; however, for some terminals at high speeds, xon/xoff flow control is required. In this case only, set *flow_lim* to a number somewhat larger than the terminal's character buffer (typically 32, 64, or 128).

In some cases, it may be desirable to set up the terminal to always use **^S/^Q** for flow control. This can be done by setting *flow_lim* to -1 (**Escape - ^X^M** flow_lim). If this is done, the ability to type **^S** or **^Q** from the keyboard is lost, and they must be reset or *controlify* mode used to send them.

no_break Suppress Breaks (ON/**OFF**) □ When this parameter is on, EMACS disables the special handling of the break key to interrupt commands in progress. It may be useful for noisy lines that tend to generate breaks.

4.5.6 SPECIFYING DEFAULT MODES FOR A FILE

The modes to be used while editing a particular file can be specified by putting the string "EMACS_MODES: " somewhere in the first 10 lines of the file. The text on the same line following EMACS_MODES: is taken as names of modes to set on or off. A mode name preceded by **!** is set off, while mode names just listed are set on. If **=** immediately follows a mode name, the characters immediately following the **=** are taken as the value for the mode name. Any text on the line that does not correspond to a mode name is ignored. Therefore, the following line in a c source file sets *c* mode on, *fill* mode off, and the column for starting comments to 43.

/* EMACS_MODES: !fill c, comcol=43 */

These modes are set whenever the file is read and whenever buffers are switched.

4.6 GETTING STARTED

4.6.1 GENERAL

When EMACS is invoked, several operations are performed to set up an editing session. These include determining the type of terminal, processing an initialization file that allows customizing EMACS, and processing command line arguments.

4.6.2 TERMINAL TYPE

EMACS supports only VT100 compatible terminals. The VT100 compatible terminals include the Lucent 5425 and the 4425 terminals. Baud rate should not exceed 4800~baud and slow-scroll mode must be turned off.

4.6.3 INITIALIZATION FILE

When started, EMACS consults the initialization file to initialize the various modes to the user specifications. The initialization file is called ".emacs_init" and is located in the user home directory. The file is read as a keyboard macro, and thus should contain *exactly* what the user would type from the terminal in order to set it up. See the following example:

^U^X^Mfill

^X^Mc
(Sets c mode and unsets fill mode)

If the user does not have an *.emacs_init* file, EMACS runs the standard initialization file. The standard initialization file is located at *~EMACS/.emacs_init*. EMACS does not run both by default. However, the standard initialization file can be designated to run from the user file by placing *^X^I~EMACS/.emacs_init* in the user file at the point the system initialization is desired. Command line arguments can also be used to specify alternate initialization files as described in Section 4.6.4 .

4.6.4 COMMAND LINE ARGUMENTS

EMACS accepts several arguments on the command line that effect processing. These arguments are processed in order until an argument that is not recognized as a command line option is found. The first such argument is treated as the file to read, and any subsequent arguments can be accessed by the user using the **^X^A** command. This allows arguments following the filename to be treated in any way desired, specifying more options or filenames as interpreted by the user. The following are the supported options:

-i	(init file) EMACS interprets the next argument as the name of an initialization file to run in addition to the standard initialization file (the user init file or the system init file). The specified file is run *after* .emacs_init.
.i	(init file) EMACS interprets the next argument as the name of an initialization file to run *instead* of the standard file. This option must be the first argument to have this effect.
+n	After the specified file is accessed, EMACS moves to line n. This argument must be the last of the options, immediately prior to the filename.
<filename>	The last argument on the command line should be the name of the file to be edited. This file (if present) is read into the buffer Main. If no filename is specified, EMACS puts the user in the buffer Main with no associated file.

4.6.5 HELPFUL HINTS

The following are helpful hints for using EMACS.

(1) Learn a few of the basic commands at a time. A lot can be accomplished with just the basic commands.

(2) EMACS tries to be reasonably efficient about the refreshing of the screen. However, some sequences cause lots of text to be redisplayed. When a lot of text is to be typed into the buffer, it is recommended that blank space be opened by using **^O** then type in changes. Once the changes and additions are complete, kill any unneeded blank lines.

(3) Use **M-?** when in doubt about a command. The explanations are brief but should be sufficient.

(4) Like most editors, EMACS maintains a local buffer, so that changes made do not go into the file until the next write. Type **^X^S** reasonably frequently so as to avoid being wiped out by machine crashes, editor bugs, or other unpredictable events. If set, *save* mode does this automatically.

(5) Characters from other programs that are sent to a terminal that is running EMACS causes EMACS to become confused. (For example, output from a "background" program or text from a write command.) If the user suspects that the display does not correspond to the buffer that is being edited, type **^L** to refresh the screen. After typing **^L**, the screen matches the buffer being edited.

4.6.6 LIMITATIONS OF THE EDITOR

The following limitations may be encountered when using the EMACS editor:

- The maximum length of an EMACS editor line is 511 characters. Lines longer than 511 characters are truncated.
- The maximum number of buffers is 12.
- The kill stack contains the 16 most recent deletions, or a total of 256K characters.
- Filenames are limited to 128 characters.
- The undo command removes only about 10 distinct replacements done in one replace command.

4.6.7 RECOVERING FROM VARIOUS PROBLEMS

Due to EMACS putting the terminal in the "raw" mode, the response to interrupt and quit characters is not the same as other *UNIX*® RTR operating system programs. EMACS can usually be interrupted with a break command (**^Z** to exit from EMACS in response to the error message). In some instances, a kill may be done from another terminal or logout.

EMACS tries to save the buffers if there are internal problems or if the user does a kill or logout while EMACS is running. The buffers are saved in the files emacs0-emacs11 in the home directory. Mail, describing the files received from EMACS, is received. If these files are not cleaned up, they continue to reappear each time a logout or a kill occurs during an EMACs process.

4.7 CONCLUSIONS

The EMACS editor provides an effective means of using a high-speed display terminal for text editing. EMACS provides a variety of unique features that are very useful in editing. User reaction indicates that EMACS has improved their productivity; however, there are no quantitative measurements of this effect.

EMACS uses more computing resources than the standard *UNIX*® RTR operating system editor; however, the resources utilized do not appear to be a serious problem.

```
#include <stdio.h>
/* EMACS_MODES: c, !fill, comcol=43 */


/* This is a c program */

main()
{
        int i;
         char c;

         for (i = 0;i > 0; i++) {
                 printf("i = %d\n",i); /* print i */
         }
 }


EMACS 4.8  (0) Main > test.c
```

Figure 4-1 EMACS Screen Display

Table 4-1 EMACS Command Summary

COMMAND	DEFINITION
^@	Sets the mark at the cursor position
^A	Moves to the beginning of the line
^B	Moves back one character
^C	Capitalizes the current character
^D	Deletes current character
^E	Moves to the end of the line
^F	Moves forward one character
^G	Quits from any command in progress
^H	Deletes backward one character
^I	Inserts a tab
^J	Opens a new line and moves to the beginning of it if the next line is non-empty; otherwise, moves down one line
^K	Kills to end of line (with argument, kills multiple lines)
^L	Refreshes the screen
^M	Opens a new line and moves to the beginning of it if the next line is non-empty; otherwise, moves down one line
^N	Moves down one line
^O	Opens up a new line
^P	Moves up one line
^Q	Quotes the next character
^R	Starts a reverse search
^S	Starts a search
^T	Transposes the next two characters
^U	Multiplies the argument by four
^V	Moves to the next page
^W	Kills the current region (between cursor and mark)
^X	Is a prefix for more single character commands
^Y	Restores last killed text (leaves cursor and mark around it)
^Z	Exits one level
^[	Makes the next character a meta character
^]	Makes a local variable of a macro invocation the argument to the next command
^^	Causes the last returned result to become the argument (space); is self-inserting, and checks for automatic word wrapping
#	Is self-inserting
-	Is self-inserting unless part of a numeric argument
.	Is self-inserting and check for automatic word wrapping
0	Is self-inserting unless part of a numeric argument
1	Is self-inserting unless part of a numeric argument
2	Is self-inserting unless part of a numeric argument
3	Is self-inserting unless part of a numeric argument
4	Is self-inserting unless part of a numeric argument
5	Is self-inserting unless part of a numeric argument
6	Is self-inserting unless part of a numeric argument
7	Is self-inserting unless part of a numeric argument
8	Is self-inserting unless part of a numeric argument
9	Is self-inserting unless part of a numeric argument
:	Is self-inserting, and readjusts indentation in C mode
}	Is self-inserting, and readjusts indentation in C mode
^?	Deletes backward one character
M-^H	Deletes the last word
M-^L	Redisplays with current line at top of page
M-^M	Mails the current buffer
M-^Q	Returns the next input character (in a macro)
M-^R	Regular expression query replace
M-^S	Regular expression search
M-^X	Executes argument 0 as a character command.
M-^]	Assigns the result of the next command to a macro local variable
M-	(Meta space) Sets the mark at the cursor position
M-!	Gets and executes a shell command
M-"	Auto fills the whole buffer
M-$	Executes a command, saving the output in buffer .exec
M- -	Is self-inserting unless part of a numeric argument
M-/	Starts a comment
M-0	Is self-inserting unless part of a numeric argument
M-1	Is self-inserting unless part of a numeric argument
M-2	Is self-inserting unless part of a numeric argument
M-3	Is self-inserting unless part of a numeric argument

M-4	Is self-inserting unless part of a numeric argument
M-5	Is self-inserting unless part of a numeric argument
M-6	Is self-inserting unless part of a numeric argument
M-7	Is self-inserting unless part of a numeric argument
M-8	Is self-inserting unless part of a numeric argument
M-9	Is self-inserting unless part of a numeric argument
M-:	Maps a character to a command
M-<	Moves to top of file
M->	Moves to bottom of file
M-?	Explains the next character
M-E	Expands an environment variable and returns the result on the kill stack
M-X	Calls a macro by name
M-\	Converts its argument to a character and inserts it
M-_	Underlines the next word
M-a	Moves to beginning of sentence
M-b	Moves back one word
M-c	Capitalizes the next word
M-d	Deletes the next word
M-e	Moves to end of sentence
M-f	Moves forward one word
M-g	Moves to a specific line (its argument)
M-l	Converts the next letter to lowercase
M-m	Displays active modes
M-p	Puts the current region in the kill buffer without killing it
M-q	Quotes the next character and adds the 0200 bit
M-r	Starts query replace
M-s	Gives EMACS statistics
M-u	Undoes the last significant text modification
M-v	Moves back one page
M-w	Puts a wall chart of explanations in the buffer
M-x	Calls a macro by name
M-y	Replaces the last restore (^Y) with the next text in the kill stack
M-{	Enters a command sequence (in a macro)
M-}	Exits one level
M-~	Marks a buffer as being unmodified (up to date)
M-^?	Deletes the last word
Control-X Commands	
^X^A	Accesses the argument list to EMACS
^X^B	Changes buffers (Change to * lists active buffers)
^X^C	Exits gracefully (after asking whether or not to save the buffer)
^X^D	Changes the working directory
^X^E	Calls EMACS recursively taking input from the terminal
^X^F	Edits a file in its own buffer (if file has been read into a buffer, moves to it)
^X^I	Redirects input from a file
^X^K	Kills a buffer
^X^L	Loads a file full of macro definitions
^X^M	Sets mode from argument (prompts for mode name) and string if necessary
^X^N	Changes the buffer or filename
^X^O	Switches between windows
^X^Q	Returns the character under the cursor (in a macro)
^X^R	Reads a new file
^X^S	Saves the buffer in the current file (if modified)
^X^T	Prompts for a buffer name and inserts the text between the cursor and the mark into the named buffer
^X^U	Updates the display and delays for a specified time
^X^V	Puts the current version on the kill stack
^X^W	Writes a new or old file
^X^X	Exchanges the mark and the cursor
^X^^	Causes the current window to grow by one line
^X!	Begins a case statement (in a macro)
^X#	Reads or writes global variables
^X%	Exchanges the top of the kill stack with another item
^X&	Compares two strings
^X(	Starts a keyboard macro
^X)	Ends a keyboard macro
^X+	Causes the next entry to the kill stack to append to the previous entry
^X-	Pops the kill stack
^X1	Exits two-window mode
^X2	Enters two-window mode
^X<	Pushes a string from the TTY or macro text into the kill stack
^X=	Gives statistics about the buffer
^X>	Duplicates an item on the kill stack
^X@	Prompts the user with a string from the kill stack and returns the result
^XB	Puts the buffer name into the kill stack
^XE	Executes the keyboard macro
^XF	Puts the filename into the stack

^XL	Makes the next character a meta character
^XT	Traces the next command
^X^	Enters a "while" loop (in a macro)
^Xd	Defines macros from the current buffer
^Xg	Moves to a screen position (arg=128*y+x);
^Xm	Sets mode from argument (prompts for mode name) and string if necessary
^X|	Begins a conditional execution sequence (in a macro)
^X~	Performs arithmetic or logical operations (in a macro)

5. COMMANDS

This section describes programs invoked directly by the user or by command language procedures as opposed to subroutines that are intended to be called by the program of the user.

This section consists of many independent entries. The name of the entry appears in the first line of the text, in alphabetical order. Some entries may describe several routines, commands, etc. In such cases, the entry appears only once, alphabetized under its "major" name.

ADMIN

***** See Warning in Section 1.1 *****

1. NAME

admin - Administer logins

2. SYNOPSIS

admin -a *nlogin* [**-u|p**] [**-s** *shell*] [**-h** *home*]

admin -a *nlogin-* [**-r|R**]

admin -d *ologin*

admin -c *ologin*

admin -o *ologin* **-n** *nlogin*

3. DESCRIPTION

Admin allows user logins to be added, deleted, and renamed.

Admin also allows passwords to be added or deleted for password-protected commands. In addition, *admin* permits the password of any login to be cleared. Only the super user (root) and the user administrator (manager) may invoke this command.

3.1 Creating a User Login

The format for creating a user login is as follows:

admin -a *nlogin* **-u**|**p** [**-s** *shell*] [**-h** *home*]
*Where **nlogin** is the desired new login.*

When a new login is created, a new entry is made in the password file with the desired name. The uid is unique and not within the range of system logins. The group id is the same for all new creations. This is controlled by the #define CFGID (10) in the source. The password is cleared. Password aging is set for 2 weeks minimum and 10 weeks maximum. A login directory is created in the directory specified by the #define of CFTDIR (/unixa/users) in the source. If the login is for the *UNIX*® operating system use, "-u" must be specified. If the login is for PDS/MML use, "-p" must be specified. An option login shell and home directory may also be specified for special applications.

3.2 Creating a Command Password

The format for creating a command password is as follows:

admin -a *nlogin-* [**-r**|**R**]
*Where **nlogin-** is the desired new login.*

This new login must be the name of a command to be password-protected with a tailing "-" dash. Thus, to password protect the command **sh**, the *nlogin-* should be **sh-**.

When a new login is created, a new entry is made in the password file with the desired name. The uid is unique and within the range of system logins. The group id is the same for all creations. This is controlled by the #define CFGID (10) in the source. The password is cleared. Password aging is set for 2 weeks minimum and 10 weeks maximum. If "**-r**" is specified, the command may have limited access (only a few commands have a limited access mode). If "**-R**" is specified, the command will have full access.

The *passwd* command can then be used with this *nlogin-* to set the password of the command.

3.3 Deleting a Login

The format for deleting a login is as follows:

```
admin -d ologin-
  Where ologin is the login to delete.
```

System logins cannot be deleted. Only craft and command logins can be deleted. This procedure deletes all at(1) and crontab(1) jobs owned by the login. If the login home directory ends [right after a slash ('/')] with the login name, then the login home directory and all files within it are deleted. Finally, the entry from the password file is deleted.

3.4 Renaming a Login

The format for renaming a login is as follows:

```
admin -o ologin-  -n nlogin
  Where ologin is the old login and nlogin is the new login name.
```

System logins cannot be renamed. The procedure is to delete all at(1) and crontab(1) jobs owned by the old login.

If the old login home directory ends [right after a slash ("/")] with the old login name, then the new home directory is the same as old except that the new login name replaces the old login name in the directory name.

In this case, the old home directory is renamed to be the new home directory. If the old home directory does not end with the old login name, then the home directory is not changed. No files are deleted in either case.

3.5 Clearing a Password

The format to clear a password is as follows:

```
admin -c ologin-
  Where the login password of ologin is to be cleared.
```

The clear procedure deletes the selected login password from the password file. The next time a login is attempted on this login name, a new password will be requested.

4. SYSTEM LOGINS

A system login is a login with a uid of less than or equal to the #define LIMSID (10). System logins can only have their password cleared by this command.

5. EXAMPLES

To add the pds/mml login *ralph* enter the following:

```
admin -a ralph -p
```

To add the *UNIX*® system login *bill* enter as follows:

```
admin -a bill -u
```

To clear the password of login *mary* enter the following:

```
admin -c mary
```

To rename the login *mary* as *ann* enter the following:

```
admin -o mary -n ann
```

To delete the login *richard* enter the following:

```
admin -d richard
```

To allow a password on the command *sh* enter the following:

```
admin -a sh- -r
```

6. FILE

/etc/passwd

7. SEE ALSO

at(1), crontab(1), passwd(1)

8. DIAGNOSTICS

Various diagnostics are issued if the login name is too short, too long, or wrong character set. Also, diagnostics are issued if *nlogin* already exists, or if *ologin* does not exit, or is a system login.

AT

***** See Warning in Section 1.1 *****

1. NAME

at, batch - Execute commands at a later time

2. SYNOPSIS

at *time*
 [*date*] [**+** *increment*]

at *-r* job ...

at *-l* [*job ...*]

batch

3. DESCRIPTION

At and *batch* read commands from standard input are to be executed at a later time. *At* allows you to specify when the commands should be executed, while jobs queued with *batch* will execute when system load level permits. *At* **-r** removes jobs previously scheduled with *at*. The **-l** option reports all jobs scheduled for the invoking user.

Standard output and standard error output are mailed to the user unless they are redirected elsewhere. The shell environment variables, current directory, and umask are retained when the commands are executed. Open file descriptors, traps, and priority are lost.

Users are permitted to use *at* if their name appears in the file **/unixa/lib/cron/at.allow** . If that file does not exist, the file **/unixa/lib/cron/at.deny** is checked to determine if the user should be denied access to *at.* If neither file exists, only root is allowed to submit a job. If either file is **at.deny**, global usage is permitted. The **allow/deny** files consist of one user name per line.

The *time* may be specified as 1, 2, or 4 digits. One- and two-digit numbers are taken to be hours; four digits to be hours and minutes. The time may alternately be specified as two numbers separated by a colon, meaning *hour : minute .* A suffix **am** or **pm** may be appended; otherwise, a 24-hour clock time is understood. The suffix **zulu** may be used to indicate GMT (Greenwich mean time). The special names **noon, midnight, now**, and **next** are also recognized.

An optional *date* may be specified as either a month name followed by a day number (and possibly year number preceded by an optional comma) or a day of the week (fully spelled or abbreviated to three characters). Two special "days," **today** and **tomorrow**, are recognized. If no *date* is given, **today** is assumed if the given hour is greater than the current hour and **tomorrow** is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.

The optional *increment* is simply a number suffixed by one of the following: **minutes, hours, days, weeks, months,** or **years**. (The singular form is also accepted.)

Thus, legitimate commands include the following:

```
at 0815am Jan 24
at 8:15am Jan 24
at now + 1 day
at 5pm Friday
```

At and *batch* write the job number and schedule time to standard error.

Batch submits a batch job. It is almost equivalent to "at now." It is different in the following ways:

(1) It goes into a different queue.

(2) "At now" responds with the error message too late.

At **-r** removes jobs previously scheduled by *at* or *batch*. The job number is the number given previously by the *at* or *batch* command. Job numbers can also be gotten by typing *at* **-l**. Only the super user can remove jobs of other users.

4. EXAMPLES

The *at* and *batch* commands read from standard input the commands to be executed at a later time. *Sh* (1) provides different ways of specifying standard input. Within your commands, it may be useful to redirect standard output.

The following sequence can be used at a terminal:

batch

sort *filename* > *outfile*

<control-D>
(Hold down the **control** key and press "D".)

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a shell procedure (the sequence of output redirection specifications is significant):

batch <<!

sort *filename* **2>&1** > *outfile* **| mail** *loginid* **!**

To have a job reschedule itself, invoke *at* from within the shell procedure by including a code, similar to the following, within the shell file:

echo "sh *shellfile***" | at 1900 thursday next week**

5. FILES

/unixa/lib/cron -	main cron directory
/unixa/lib/cron/at.allow -	list of allowed users
/unixa/lib/cron/at.deny -	list of denied users
/unixa/lib/cron/queue -	scheduling information
/unixa/spool/cron/atjobs -	spool area

6. SEE ALSO

cron(1), kill(1), mail(1), nice(1), ps(1), sh(1)

7. DIAGNOSTICS

Complains about various syntax errors and times out of range.

ATOMSW

***** See Warning in Section 1.1 *****

1. NAME

atomsw - Atomic switch two files

2. SYNOPSIS

atomsw file1 file2

3. DESCRIPTION

Atomsw switches the contents, permissions, and owners of two files in a single operation. In case of a system fault during the operation of this command, *file2* either has its original contents, permissions and owner, or the contents of *file1*, permissions and owner. Thus, *file2* is considered precious. *File1* may be truncated in case of a system fault.

4. RESTRICTIONS

Both files must exist. Both files must reside on the same file system. Neither file may be a "special device" (for example, a TTY port).

To enter this command from the craft-shell, switching file "/tmp/abc" with file "/tmp/xyz", enter the following applicable commands.

For MML:

```
EXC:ENVIR:UPROC,FN="/bin/atomsw",ARGS="/tmp/abc"-"/tmp/xyz";
```

For PDS:

```
EXC:ENVIR:UPROC,FN"/bin/atomsw",ARGS("/tmp/abc","/tmp/xyz")!
```

5. NOTE

File 1 may be lost during a system fault.

6. FILES

/bin/atomsw

AWK

Software Release:
***** See Warning in Section 1.1 *****

1. NAME

awk - 1985 version of awk language

2. SYNOPSIS

awk [**-F***regex*] ['program'] [parameters] [files]

awk [**-F***regex*] [**-f** 'programfile'] [parameters] [files]

3. DESCRIPTION

The 1985 *awk* command is an expanded version of the previous standard *awk* pattern-scanning and processing language. The language has been augmented by the addition of dynamically-defined regular expressions, user-defined functions, improved input/output flexibility, and new built-in functions.

The *awk* command scans each input *file* for lines that match any of a set of patterns specified in *program*. With each pattern in *program* there can be an associated action that is performed when a line of a file matches the pattern. The set of patterns may appear literally as the *program* or in a file specified as **-f** *programfile*. The *program* string should be enclosed in single quotes (') to protect it from the shell.

Parameters, in the form x=... y=... etc., may be passed to *awk*.

Files are read in order; if there are no *files*, the standard input is read. The filename **-** means the standard input. Each line is matched against the pattern portion of every pattern-action statement; the associated action is performed for each matched pattern.

An input line is made up of fields separated by white space (tabs and blanks). (This default can be changed by using FS or the **-F** parameter; see the following). The fields are denoted $1, $2 ...; $0 refers to the entire line.

A pattern-action statement has the following form:
   pattern { action }

A missing action means print the line; a missing pattern always matches.

A pattern can be one of the following:

BEGIN	Matches before any input is read
END	Matches after all input is read
relational expr	Matches if the relation holds (for example, NR == 3)
/*reg exp*/ or *reg exp*	Matches the regular expression
pattern1 && *pattern2*	Matches if both patterns match
pattern1 || *pattern2*	Matches if either pattern matches
(*pattern*)	Grouping: matches *pattern*
!*pattern*	Matches if *pattern* does not match (NOT operator)
pattern1, *pattern*	Matches from the line where *pattern1* matched up to (andincluding) the line where *pattern2* matched
func *name* (*parameter list*) { *statement* }	Defines a function called *name*.

An action is a sequence of statements. Statements are terminated by semicolons, new-lines, or right braces. Statements include the following that affect control-flow:
  if (*conditional*) *statement*

  if (*conditional*) *statement* else *statement* if (*subscript in array*) *statement* else *statement*

while (*conditional*) *statement*

for (*expression* ; *expression* ; *expression*) *statement*

break

continue

{ [*statement*] ... }

next

exit

exit *expression*

function-name (*expr, expr, ...*)

return

return *expression*

Input-output statements include the following:

close (*filename*)	Close file
getline	Set $0 from next input record
getline<" *file*"	Set $0 from next record of *file*
getline *var*	Set *var* from next input record
getline *var* <"*file*"	Set *var* from next record of *file*
print	Print current record
print *expression-list*	Print expressions
print *expression-list* >"*file*"	Print expressions on *file*
printf *format* , *expression-list*	Format and print
printf *format* , *expression-list*>"*file*"	Format and print on *file*
system(*cmd-line*)	Execute command *cmd-line*; return its exit status.

Expressions take on string or numeric values as appropriate, and are built using the operators (in increasing precedence) *blank* (string concatenation), **+/-** (addition/subtraction), ***** or **/** or **%** (multiplication/division/modulus), **++** or **--** (increment/decrement, prefix and postfix), and **$** (field value). The C operators **+=**, **- =**, **/=**, and **%=** are also available expressions, sharing the lowest precedence with the assignment operator **=**. Logical OR (**| |**), and logical AND (**&&**), regular expression match **( )** and non-match (**!**), and relational operators (**<**, **<=**, **>**, **>=**, **==**, **!=**) are also available.

Variables may be scalars, array elements (denoted x[i]) or fields. Variables are initialized to the null string (treated as zero numerically). Array subscripts may be any string, not necessarily numeric; this allows for a form of associative memory. String constants are quoted (**"**).

The printf format conversions that are supported are as follows:

%c	ASCII character
%d	Decimal number
%o	Unsigned octal number
%s	String
%x	Unsigned hexadecimal number
%%	Print a %; no argument is converted .

Additional parameters may lie between the % and the control letter:

-	Left-justify expression in its field
width	Pad field to this width as needed; leading 0 *pads with zeros* .

The following are the built-in functions:

gsub(*regex*, *string*, *target*) Substitute *string* for each substring matching regular expression *regex* in string *target*, return number of substitutions. If *target* is omitted, use $0. All occurrences of *&* in *string* are replaced by the substring matched by *regex*. The special meaning of *&* may be turned off by preceding it with a backslash, as in \*&*.

index(*target* , *string*) Return index of *string* in string *target*, or 0 if not present. (the first character is at index 1.)

length(*string*) Return length of *string*

match(*string*, *regex*) Return index of where *string* matches *regex* or 0 if there is no match; set RSTART and RLENGTH.

split(*string*, *array*, *regex*) Split *string* into *array* on regular expression *regex*, return number of fields. If *regex* is omitted, FS is used in its place.

sprintf(*fmt*, *expr-list*) Format *expr-list* according to *fmt*, and return the resulting string

sub(*regex*, *string*, *target*) Like gsub except only the first matching substring is replaced

substr(*string*, *index*, *number*) Return *number*-character substring of *string* starting at *index*. (The first character is at index 1.) If *number* is omitted, the substring goes to the end of *string*.

Patterns are regular expressions made of the following pieces (increasing precedence):

c	Match non-metacharacter c
\c	Match any literal character c
\.	Matches any character but newline
^	Matches beginning of line or string
$	Matched end of line of string
[abc...]	Character class matches any one of abc...
[^abc...]	Negated character class matches any but one of abc... and newline
r1|r2	Alternation; matches either r1 or r2
r1r2	Concatenation; matches r1, then r2
r+	Matches one or more r's
r*	Matches zero or more r's
r?	Matches zero or one r
(r)	Grouping; matches r

Regular expression constants must be surrounded by slashes; dynamic regular strings may be the values of variables or strings.

A pattern *regex* may be used to specify the field separator(s) by starting the program with the following:

BEGIN **{ FS =** *regex* **}**

or
-F *regex* option.

Other variable names with special meanings include the following:

ARGC	Number of command-line arguments
ARGV	Array of command-line arguments (0..ARGC-1)
FILENAME	The name of the current input file;
FNR	Input record number in current file
NF	The number of fields in the current record;
NR	The ordinal number of the current record;
OFMT	The output format for numbers;
OFS	The output field separator (default blank);
ORS	The output record separator (default new-line);
RLENGTH	Length of string matched by regular expression in match ()
RS	The input record separator (default new-line);
RSTART	Beginning position of string matched in match ()
SUBSEP	Separator for array subscripts of form [*i, j*,...] (default "\034")

4. EXAMPLES

Print lines longer than 72 characters:

```
length > 72
```

Print first two fields in opposite order:

```
{ print $2, $1 }
```

Add up first column, print sum and average:

```
{ s =+ $1 }

END   { print "sum is", s, "average is", s/NR }
```

Print fields in reverse order:

```
   { for (i = NF; i > 0; --i) print $i }
```

Print all lines between start/stop pairs:

```
   /start/, /stop/
```

Print all lines whose first field is different from previous one:

```
   $1 != prev { print; prev = $1 }
```

Print file, filling in page numbers starting at 5:

```
   /Page/ { $2 = n++; }
   { print }
   command line: awk -f program n=5 input
```

5. SEE ALSO

The Awk Programming Language by A. V. Aho, B. W. Kernighan, P. J. Weinberger. Published by Addison-Wesley, 1988, ISBN 0-201-07981-X.

6. LIMITATIONS

Input white space is not preserved on output if fields are involved.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add 0; to force an expression to be treated as a string, concatenate the null string ("").

The *5ESS*® switch version does not support floating point numbers nor any floating point functions.

BANNER

***** See Warning in Section 1.1 *****

1. NAME

banner - Make posters

2. SYNOPSIS

banner *strings*

3. DESCRIPTION

Banner prints its arguments (each up to 10 characters long) in large letters on the standard output.

4. SEE ALSO

echo(1)

BASENAME

***** See Warning in Section 1.1 *****

1. NAME

basename, dirname - Deliver portions of pathnames

2. SYNOPSIS

basename *string [suffix]***dirname** *string*

3. DESCRIPTION

Basename deletes any prefix ending in a slash (/) and the *suffix* (if present in **string**, and prints the results on the standard output. It is normally used inside substitution marks (") within shell procedures.

Dirname delivers all but the last level of the pathname in *string*.

4. EXAMPLES

The following example, invoked with the argument /usr/src/cmd/cat.c, compiles the named file and moves the output to a file named **cat** in the current directory:

```
cc $1 mv a.out basename $1 .c
```

The following example sets the shell variable NAME to **/usr/src/cmd:**

```
NAME=dirname /usr/src/cmd/cat.c
```

5. SEE ALSO

sh(1)

6. BUGS

The *basename* of **/** is null and considered an error.

BATCH

***** See Warning in Section 1.1 *****

1. NAME

batch

3. DESCRIPTION

See *at*.

CAT

***** See Warning in Section 1.1 *****

1. NAME

cat - Concatenate and print files

2. SYNOPSIS

```
cat[ -b ] [ -s ] file ...
```

3. DESCRIPTION

Cat reads each *file* in sequence and writes it on the standard output. The following command prints the file:

```
cat file
```

The command, input as follows, concatenates the first two files and places the result in the third file.

```
cat file1 file2 > file3
```

If no input file is given or if the argument □ is encountered, *cat* reads from the standard input file. Output is buffered unless the **-u** option is specified. The **-s** option makes *cat* silent about nonexistent files. No input file may be the same as the output file unless it is a special file.

4. WARNING

Command formats, such as the following, cause the original data in *file1* to be lost; therefore, take care when using shell special characters.

```
cat file1 file2 > file1
```

5. SEE ALSO

cp(1), pr(1)

CFTSHELL

***** See Warning in Section 1.1 *****

1. NAME

cftshell - Execute craft shell

2. SYNOPSIS

cftshell command

3. DESCRIPTION

Cftshell allows the *UNIX*® operating system user to execute a single craft shell command. Control returns to the user after the command acknowledgment (for example, OK, PF). If long running, execution of the command may continue in the background.

4. EXAMPLE

```
cftshell 'OP:CLK'
```

CGREP

***** See Warning in Section 1.1 *****

1. NAME

cgrep - Shows context of matching patterns found in files

2. SYNOPSIS

cgrep [options] [-e]patterns[files]

cgrep [options]-f pattern_file[files]

3. DESCRIPTION

Cgrep provides all the features of *grep* and *fgrep*, with greatly enhanced performance (see the section on **PERFORMANCE**) along with many additional features, one of which is the ability to output the context (surrounding lines) of the matching lines. The use of *cgrep* is upward-compatible with that of *grep* or *fgrep* (using the **-F** option). *cgrep* searches files for lines matching *patterns* and normally sends to standard output matching lines, possibly with a user-specified context window. The window may be specified as a constant number of lines before and after the matching lines (the number of context lines before the matching lines and the number after the matching lines may differ); or by specifying the beginning and ending *delimiters* (these may differ); or as any combination thereof. The windows need not be delimited at the nearest occurrence of *delimiters*, as any number of matches to the beginning and ending *delimiters* may be independently specified. The lines delimiting the beginning or end of the window may independently be either included in or excluded from the window.

By default, the *patterns* and *delimiters* are taken to be limited regular *expressions* as in *grep*; however, full regular *expressions* as in *egrep* or fixed *strings* as in *fgrep* may also be used.

More than one *pattern* or *delimiter* can be specified by enclosing the entire set of *patterns* or *delimiters* within quotes and separating individual *patterns* or *delimiters* with newlines. More than one *pattern* or *delimiter* can also be specified by using the **-E** option and separating individual *patterns* or *delimiters* with '|'. More than one *pattern* can also be specified by using the **-f** option and listing the *patterns*, one per line in a file. *Patterns* can also be specified dynamically from the input itself by use of the **-t** or **+t** option. *cgrep* also provides two special-purpose options, **-R** and **-T**, for scanning ROP output.

If no *files* are specified, standard input is assumed.

cgrep allows restricting matches to whole words or phrases. The section, **WORD MATCHING**, explains this in more detail.

cgrep supports approximate matching (matching with mismatches allowed). The description of the **-A** option explains approximate matching in more detail.

Unlike *grep* or *fgrep, cgrep* allows the matching of *patterns, delimiters,* or *trail_patterns* that may span multiple lines of text through the use of literal newline characters. The section, **MULTI-LINE MATCHING,** explains this in more detail.

4. OPTIONS

The specific options recognized by *cgrep* are grouped by functional category in the sections that follow. Some options begin with a '-', others with a '+'.

Any *cgrep* option which does not take a mandatory argument may take an optional numeric argument, which defaults to **0**. The general format is *-Qn,* or *+Qn,* □ *-Q* or *+Q* represents any option that takes no argument, and *n* is a no-negative integer of one or more digits. There cannot be any white space between *Q* and *n*. The interpretation of this numerical value is dependent on the individual option. Except when explicitly documented in the succeeding paragraph, these optional numeric arguments are ignored.

The □ option may optionally be used to terminate the command line options. Each subsequent term on the command line is then interpreted as an argument rather than as an option, even if it begins with a '-' or a '+'.

5. Pattern Source Options

The following two options specify the source of *patterns* to be matched. Either option may be specified instead of the *patterns* argument. If *patterns* begin with a '-' or a '+', one of these options *MUST* be specified.

-f *pattern_file*	The *patterns* list is taken from the *pattern_file.*
-e *patterns*	Same as a *patterns* argument, but properly recognizes *patterns* that begin with a '-' or a '+'.

6. Pattern-Matching Options

The following options affect the comparisons *patterns* for determining matched lines. Comparisons with *delimiters* or *trail_patterns* (see the following legend) may also be affected. Each of these options may be set independently for *patterns, trail_patterns*, and each of the two *delimiters*. Each option applies to all subsequent arguments on the command line, until changed by a complementary option. These options do not apply to preceding arguments on the command line. For example, the **-i** option causes upper/lower case distinction to be ignored during comparison with *delimiters* only if **-i** appears *BEFORE delimiters.*

-E	Extended *grep* mode wherein *patterns, trail_patterns,* or *delimiters* are interpreted as full regular *expressions*. Extended grep mode is the default if the *cgrep* executable file is named *egrep*.
-F	The (*fgrep* mode) *patterns, trail_patterns,* or *delimiters* are interpreted as fixed *strings* as in *fgrep. fgrep* mode is the default if the *cgrep* executable file is named *fgrep.*
-G	The (*grep* mode) *patterns, trail_patterns,* or *delimiters* are interpreted as limited regular *expressions* as in *grep*. *grep* mode is the default unless the *cgrep* executable file is named *egrep* or *fgrep*.
-i	Ignore upper/lower case distinction during comparisons.
+i	(default) Comparisons are case sensitive.
-x	Matches must be made to an entire line of input. (The **-x** option does not affect comparisons with *trail_patterns.)*
-x1	Matches must be made to an entire word or phrase of input. The section, **WORD MATCHING,** explains this in more detail. (The **-x1** option does not affect comparisons with *trail_patterns.*)
+x	(default) Matches may be made to a portion of an input line or word.
-a	Allow easy specification of non-printing ASCII characters in *patterns, trail_patterns* or *delimiters,* using the backslash character with a syntax like that in C language strings. Any 8-bit ASCII character may be represented by \ddd, where *ddd* is the ASCII character code in one to three octal digits. Also, **\b, \t, \n, \v, \f, \r,** or **\a** can be used to represent the back

space, tab, newline, vertical tab, form feed, carriage return, and alarm bell characters respectively. All other uses of backslash represent the literal character following the backslash. For example, \\ represents the backslash character itself. The **-a** option affects all uses of backslash, even those inside regular expression character classes (square brackets) and even those in *fgrep* mode.

+a (default) Backslash characters are interpreted "normally," as they would be in *grep* or *fgrep*, depending on which mode is in effect.

-A (*nerrors*) Allow up to *nerrors* mismatches in matching *patterns* or *delimiters.* (Comparisons with *trail_patterns* are unaffected.) *nerrors* must be a positive integer or zero. By default, a single mismatch is either the insertion of an extra character in the text, a substitution of a character in the text, or a deletion of a character in the text. However, portions of a fixed string or regular expression that are enclosed in angle brackets (< and >) must be matched exactly.

NOTE: Angle brackets become meta-characters in all regular expressions or fixed strings for which approximate matching is used. For performance reasons, it is a good idea to use angle brackets liberally when using approximate matching. Like other meta-characters, an angle bracket may be escaped in a regular expression by preceding it with a backslash, and an angle bracket retains its literal meaning inside of regular expression character classes (square brackets). In a fixed string, an angle bracket may be escaped by preceding it with a backslash and using the **-a** option.

The cost of deletions, insertions, or substitutions may be set independently with the **-C, +C,** or **-+C** options respectively. These options are explained in succeeding paragraphs. The section, **APPROXIMATE MATCHING EXAMPLES,** gives examples of the use of approximate matching.

-C (*deletion_cost*) Sets the cost of a deletion error to *deletion_cost*, which must be a positive integer. By default, a deletion error counts as one mismatch.

+C (*insertion_cost*) Sets the cost of an insertion error to *insertion_cost*, which must be a positive integer or zero. By default, an insertion error counts as one mismatch. If *insertion_cost* is zero, approximate matching is used even if the **-A** option is not used or if zero mismatches are specified. Also, if *insertion_cost* is zero, the insertion of a new-line character still counts as one mismatch.

-+C (*substitution_cost*) Sets the cost of a substitution error to *substitution_cost*, which must be a positive integer. By default, a substitution error counts as one mismatch.

+-C (*substitution_cost*) Equivalent to **-+C** *substitution_cost*

The following options also affect the comparisons with *patterns* for determining matched lines. Comparisons with *delimiters* or *trail_patterns* are unaffected.

-v Lines that do *NOT* contain one or more *patterns* are matched.

NOTE: See also the **-V** option. The **-v** and **-V** options, although related, are quite different.

+v (default) Lines that contain one or more *patterns* are matched.

-N Match only the first *nmatches* lines containing *patterns*. Subsequent occurrences of *patterns* will not be matched.

+N (*nmatches*) Don not match the first *nmatches* lines containing *patterns*. Subsequent occurrences of *patterns* are matched.

-+N	(*nmatches*) Same as **+N** *nmatches* and **-N** *nmatches*.
+-N	*nmatches*Same as **+N** *nmatches* and **-N** *nmatches*.

7. Windowing Options

The following options cause lines surrounding each matched line to be included in a context window. If none of these options are specified, only the matched lines are included in the context windows. Lines in the context windows are normally printed to standard output in the order they appear in the input files.

-nline	In addition to the matching line, include the *nline* preceding lines. There is no maximum value for *nline*.
+nline	In addition to the matching line, include the *nline* subsequent lines. There is no maximum value for *nline*.
+-nline	Same as *+nline* and *-nline*.
-w	(*delimiters*) In addition to the matching line, include the lines preceding it. If *-nline* is not also specified, the beginning of the context window is delimited by the closest line that contains one or more *delimiters*. If *-nline* is also specified, the beginning of the context window is delimited by the *nline*th closest line that contains one or more *delimiters*. If the **-I** option is specified (the default), the delimiting line is included in the window, and the matching line is included in the search for *delimiters*. If the **-I1** option is specified, the delimiting line is included in the window, but the matching line is excluded from the search for *delimiters*. In contrast, if the **-I2** option is specified, the delimiting line is excluded from the window, and the matching line is excluded from the search for *delimiters*.
+w	(*delimiters*) In addition to the matching line, include the lines following it. If *+nline* is not also specified, the end of the context window is delimited by the closest line that contains one or more *delimiters*. If the **+I** option is specified (the default), the delimiting line is included in the window, and the matching line is included in the search for *delimiters*. If the **+I1** option is specified, the delimiting line is included in the window, but the matching line is excluded from the search for *delimiters*. In contrast, if the **+I2** option is specified, the delimiting line is excluded from the window, and the matching line is excluded from the search for *delimiters*.
-+w	(*delimiters*) Same as **+w** *delimiters* and **-w** *delimiters*.
+-w	(*delimiters*) Same as **+w** *delimiters* and **-w** *delimiters*.
-W	(*delimiters*) Same as **-w** *delimiters* except that the window is delimited by lines that do *NOT* contain one or more *delimiters*.
+W *delimiters*	Same as **+w** *delimiters* except that the window is delimited by lines that do *NOT* contain one or more *delimiters*.
-+W	(*delimiters*) Same as **+W** *delimiters* and **-W** *delimiters*.
+-W	(*delimiters*) Same as **+W** *delimiters* and **-W** *delimiters*.

The following options are used in conjunction with the windowing options described previously. These options have no effect if used without the appropriate windowing option.*The symbol* *******designates options that have been defined in previous paragraphs.*

-l	(default) **-w** and **-W**
+l	(default) **+w** and **+W**
-+l	(default) Same **-l** and **+l**
+-l	(default) Same **-l** and **+l**
-l1	**-w** and **-W**
+l1	**+w** and **+W**
-+l1	Same **-l1** and **+l1**
+-l1	Same **-l1** and **+l1**
-l2	**-w** and **-W**
+l2	**+w** and **+W**
-+l2	Same **-l2** and **+l2**
+-l2	Same **-l2** and **+l2**.

8. Output Options

The following options affect only the output of matching lines and their context windows. The actual matching of lines and the generation of context windows are unaffected by these options. By default, any line in one or more context windows is output exactly once in the order that the lines appear in the input.

-h	The output of the filename on each output line is suppressed.
-h1	The filename is always output preceding each output line.
+h	(*default*) If two or more input files are specified (and the **-M** option is not specified), the filename is output preceding each output line.
-b	Each line output is preceded by the block number on which it was found. This is sometimes useful in locating disk block numbers by context.
-b1	Each line output is preceded by the byte offset of the beginning of the line.
+b	(*default*) No block or byte number is output.
-n	Each line output is preceded by its relative line number in the file.
+n	(*default*) No line number is output.
-H	If the output is directly to a terminal, matched lines are highlighted so that they stand out from the context windows. If the output is not directly to a terminal, this option has no effect.
-H1	If the output is directly to a terminal, matched lines are highlighted so that they stand out from the context windows. If the output is not directly to a terminal, the appropriate escape sequences, for the current terminal type, are inserted into the output.
+H	(*default*) Matched lines are not highlighted.
-o	If two or more windows overlap or touch, output each window separately in its entirety. Lines common to multiple windows are output multiple times, once in each window. **-o** is ignored if no windowing option is specified.
	This option could conceivably cause the output of N squared lines from an N line file. Use cautiously when large windows are specified.
+o	(*default*) Overlapping or touching windows are output as a single merged window, so that no input lines are output more than once.
-d	(*output_separator*) Specifies an output string to be used to separate windows. If any surrounding lines are requested, a separator normally separates each printed window. The output separator defaults to a line of equal signs. This option can be used to specify a different output separator.
	No output separator is printed when no surrounding lines are requested, unless this option is specified.
-D	The default output separator between windows is suppressed.
+D	(*default*) The default output separator between windows is not suppressed.
-V	Matching lines and their context windows are *NOT* output. All other lines are output.
	The **-V** *and* **-v** *options are quite different.*
+V	(*default*) Lines in context windows are output.
-c	A count of output lines is printed (excluding output separators) rather than the lines themselves. The count is normally preceded by the filename, if there are two or more input files. The count is equal to the total number of lines that would have been printed (excluding output separators) had the **-c** option not been specified.
-c1	Same as **-c**, except that the printout is suppressed for files having zero output lines.
+c	(*default*) The output lines are printed.
-l	Only the names of files with lines to be printed (once), separated by new-lines.
-l1	No output is printed. This option is useful when only the exit status is required.
-l2	Only the names of files with *NO* lines to be printed are printed (once), separated by new-lines.
+l	(*default*) The output lines are printed.

9. Text Input Options

The following options affect the handling of input text *files.*

-s	The error messages normally output for nonexistent or unreadable files are suppressed.
-s1	The error messages normally output for nonexistent or unreadable files are suppressed, and such files do

	not affect the exit status of *cgrep.*
+s	(*default*) Error messages are output for nonexistent or unreadable files.
-M	If this option is specified, *cgrep* treats multiple input *files* as a single concatenated file. More precisely, *cgrep.* **-M** *pattern files* is equivalent to *cat files* | *cgrep pattern.*
+M	(default) If multiple *files* are specified, *cgrep* treats each file independently.
-B	(*bufsize*) Specifies the buffer size used to read input form a pipe or a special device file. *Bufsize* (default 64) is the size of the input buffer in kbytes. The buffer size may need to be decreased from the default value when *cgrep* is being used interactively with interactively generated input. An example follows: **tail -f** *file* | **cgrep -B 1** *pattern* This may need to be done because *cgrep* processes input only when an input end-of-file is encountered or the input buffer is full. However, the buffer size should not be set smaller than the size of a typical text window (see the LIMITATIONS section). *Bufsize* is used only if input is being read from a pipe or a special device file. In all other cases, *cgrep* optimizes the buffer size to hold the input files and ignores the value of *Bufsize*.

10. Dynamic Pattern-Matching Options

The following options allow the dynamic creation of patterns for matching.

-t (*trail_patterns*) If a line matching *trail_patterns* is found on a line also matching *patterns* or in the context window of a line matching *patterns*, the portion of input that matched *trail_patterns* is saved as a fixed *string* which is treated like one of the *patterns* for subsequent matches. Only the 32 most recently matched fixed strings are saved, with newly saved fixed strings replacing previously saved fixed strings, oldest first. For example, if **-t 'EVENT=[0-9]'** is specified and the string **"EVENT=6"** is found on a line also matching *patterns* or in the surrounding window of a line matching *patterns*, subsequent occurrences of the fixed string **"EVENT=6"** would generate a context window, subject to the specified limitation of 32 saved fixed strings. This process of saving fixed strings is non-recursive in the sense that matches with previously saved fixed strings do not generate new fixed strings to be saved. See the description of the **-T** option for another example of the use of the **-t** option.

+t (*trail_patterns*) This option is equivalent to **-t.**

11. *5ESS*® Switch Project-Specific Options

The following special-purpose options are provided for the convenience of those analyzing *5ESS*® Switch International or Domestic ROP reports.

-R Sets window *delimiters* and other options for *5ESS*® Switch International or Domestic ROP reports. The **-R** option can also be used in conjunction with *-nline* or *+nline* to output preceding or subsequent ROP reports to the matched report, or both. In either case, *nline*-1 adjacent reports are also printed. The **-R** option is almost equivalent to **-DEMh +I2 □+w '\+\+\+|^Y'** but has one additional effect. Patterns of the form **'sm=***n'* (the literal **sm** must be in lower case letters) are expanded into regular expressions to match most if not all ROP reports pertaining to SM *n*. The *n* must be an unsigned number without leading zeros; in which, each digit may be either a simple digit or any one of a set of digits enclosed in square brackets. Within brackets, a hyphen can be used to indicate a range of digits. See the *5ESS*® **SWITCH EXAMPLES** section for examples of the use of this feature.

-T Same as **-R,** except also causes ROP event trails to be followed. If the string **"EVENT=***n*" (*n* is a number of one or more digits, possibly preceded by a minus sign) or **"EVENTO=***n* (Spanish) appears in a matched ROP report, all subsequent reports with the same event number are printed. Equivalent to the following: **-R -t 'EVENTO?=-?[0-9]+'**.

12. Pattern-Matching Summary

In all matching modes, more than one *pattern, trail_pattern,* or *delimiters* can be specified by enclosing the entire set of *patterns, trail_patterns,* or *delimiters* within quotes and separating individual *patterns, trail_patterns,* or *delimiters* with newlines. More than one *pattern* or *delimiter* can also be specified in *egrep* mode by separating individual *pattern* or *delimiter* with '|'. More than one *pattern* can also be specified in any mode by using the **-f** option and listing the *pattern* one per line in the file.

If there is more than one input file, the filename is normally output on each line unless an option suppressing this output is specified. Care should be taken when using characters from the following set □ **{ $ * [^ | () \}**□ in *pattern, trail_patterns,* or *delimiters* because they are also meaningful to the shell. It is safest to enclose *pattern, trail_patterns,* or *delimiters* in single quotes '...'.

13. WORD MATCHING

The *cgrep* command allows the restriction of matches to complete words or phrases. The **-x1** option adds the restrictions that matched strings or regular expressions must be either at the beginning of a line or immediately following a character other than a letter, number, or underscore, and also must be either at the end of a line or immediately before a character other than a letter, number, or underscore. The **-x1** option does not affect comparisons with *trail_patterns.*

There is a second method to restrict matches to complete words or phrases. In UNIX System V Release 4 (SVR4), limited regular expressions as used by *grep(1)* have been enhanced by recognizing '\<' and '\>' as begin-word and end-word delimiters, respectively. Similarly, *cgrep,* in *grep* mode, recognizes these delimiters. The begin-word and end-word delimiters are used as follows:

cgrep *'\<RE' files* Matches the regular expression RE either at the beginning of a line or immediately following a character other than a letter, number, or underscore.

cgrep *'\>RE' files* Matches the regular expression RE either at the end of a line or immediately before a character other than a letter, number, or underscore.

cgrep *'\<RE\>' files* Matches the regular expression RE only if both of the previous conditions hold. Unlike *grep(1),* in UNIX SVR4, *cgrep* recognizes '\<' and '\>' only if used at the beginning or end of a regular expression, respectively.

14. MULTI-LINE MATCHING

cgrep is capable of matching *patterns, delimiters,* or *trail_patterns* that span multiple lines of input text. No special option is needed. Inside character classes (square brackets) of regular expressions, newline characters are interpreted as literal characters. In regular expressions, outside of character classes, newline characters must be escaped with a backslash to be interpreted literally rather than as separators between multiple expressions. Also, if the **-a** option is used, **'\n'** can be used anywhere in any mode to represent a literal newline character. *cgrep* treats literal newlines as begin-line or end-line metacharacters. A literal newline character matches the beginning or the end of any line, even if no newline is present in the text.

There is one special rule for newlines in complementary character classes, that is, character classes where the first character after the opening square bracket is a carat (^). In **grep,** or **egrep,** a complementary character class matches any character NOT in the set of listed characters, except that it does not match newlines. For compatibility, **cgrep** handles complementary character classes similarly. However, in **cgrep,** if the character set specified in a complementary character class includes a newline, newlines are matched. See the following example.

```
cgrep -a '[0-9]' file
```

[*Matches all characters except decimal digits and newlines (as in* **grep** *or* **egrep**)].

```
cgrep -a '[0-9\n]' file
  (Matches all characters except decimal digits, including newlines.)
```

15. EXAMPLES

The following command line prints each line in *file1* and *file2* that contains the string "dog" or the string "cat", and also prints the preceding three lines and the subsequent five lines:

```
cgrep -3 +5 -E 'dog|cat'file1 file2
```

The following example prints each line from the standard input that contains the string "dog", and also prints the preceding lines beginning with a line containing "heckle" and the subsequent lines ending with a line containing "jeckle":

```
cat files | cgrep -w heckle +w jeckle dog
```

Either of the following examples prints each line containing the literal **REPT** and also prints the surrounding lines until a line consisting of just **^Y** (*CTRL*-**Y**) is encountered before and after **REPT**:

```
cgrep -x -+w ^Y +x REPT file
                  or
cgrep -+w '^^Y$' REPT file
```

The following example outputs each line containing "this" or "that" and also outputs a surrounding window beginning at a line containing the fixed string "/*" and terminating at a line containing a capital letter as the first character in the line:

```
cgrep -F -w '/*' -G +w '^[A-Z]' -E 'this|that' file
```

Either of the following examples prints all paragraphs containing "keyword" in the *nroff* source file *doc*:

```
cgrep -F -x -+w .P +x keyword doc
                  or
cgrep -+w '^\.P$' keyword doc
```

Either of the following examples prints all paragraphs *NOT* containing "keyword" in the *nroff* source file *doc*:

```
cgrep -V -x -F -+w .P +x keyword doc
                  or
cgrep -V -+w '^\.P$' keyword doc
```

Either of the following examples prints all paragraphs containing a line *NOT* containing "keyword" in the *nroff* source file *doc*:

```
cgrep -v -x -F -+w .P +x keyword doc
                  or
cgrep -v -+w '^\.P$' keyword doc
```

Either of the following examples prints each line that contains one or more of the strings "Larry", "Curley", or "Moe":

```
cgrep 'Larry
  Curley
  Moe' file
                  or
cgrep -E 'Larry|Curly|Moe' file
```

The following example prints each line containing one or more of the strings "Larry", "Barry", or "Harry".

```
cgrep '[LBH]arry' file
```

The following example prints each line that contains "Hello, world", and also prints the preceding four lines and each subsequent line until the third subsequent line containing one or more of the strings "This is it" or "That is it" is found:

```
cgrep -E -4 +3 +w "This is it|That is it' 'Hello, world' file
```

The following example prints the fourth and fifth line containing "keyword" in the file *doc:*

```
cgrep +N 3 -N 2 keyword doc
```

16. APPROXIMATE MATCHING EXAMPLES

The following example prints all paragraphs in an *nroff* source file that contain (among others) the strings "Connecticut", "Connecticut", "Connecticut", and "Connecticut". It does not match "Connecticut", or "Connectickut", because "cut" must be matched exactly. It also does not match "Conetacut", because three errors is one error too many.

```
cgrep -+w '^\.P$' -A2 '<Con>necti<cut>' file
```

NOTE: The -A option flag must appear AFTER the delimiter '^\.P$' or else the approximate matching would apply to the paragraph delimiter as well.

The following example prints each line that contains (among others) the strings "Connecticut", "Connectickut", and "Connetacut". It does not match "Conneticut", because the cost of a deletion error, four, is too high. It also does not match "Connectakut", because the total cost of two substitution errors, four, is too high. It also does not match "Colnnecticut", because "Con" must be matched exactly, even though the cost of an insertion is zero.

```
cgrep -A3 +C0 -C4 +-C2 '<Con>necticut' file
```

Either of the following examples prints each line that contains no more than three non-numeric characters, and any number of numeric characters.

```
cgrep -A3 '^[0-9]*$' file
          or
cgrep -x -A3 '[0-9]*' file
```

17. *5ESS*® SWITCH EXAMPLES

The following example prints all ROP reports containing the string "CLNK" and the event trails following these reports:

```
cgrep -T CLNK ropfiles
```

The following example prints all ROP reports containing the string "CLNK" and also the five preceding and two subsequent reports:

```
cgrep -R -6
```

The following example prints all ROP reports containing the string "CLNK" and also the five preceding and two subsequent reports, provided that they are in the same file as the matched report. The **+M** option must appear *AFTER* the **-R** option in order to override the **-M** option implied by **-R**:

```
cgrep -R +M -6 +3 CLNK ropfiles
```

The following example prints all ROP reports containing the string "CLNK" and also all ROP reports pertaining to SM 12,21,24,25,26, or 27:

```
cgrep -R 'CLNK|sm=12|sm=2[14-7]' ropfiles
```

18. SEE ALSO

ed(1),fgrep(1),grep(1),sed(1),sh(1), in document 235-700-200 *RTR UNIX Manual.*

19. DIAGNOSTICS

Normally, the exit status is **0** if any matches are found; **1** if no matches are found, and **2** for syntax errors or inaccessible files (even if matches were found). However, if the **-s1** option is specified, inaccessible files do not affect the exit status. Also, if the **-V** option is specified, the exit status is **0** only if there are any lines that are outside of all context windows of matched lines.

20. PERFORMANCE

cgrep is far superior in performance to the current versions of *grep* or *fgrep*. *cgrep* should be used in place of *grep* or *fgrep* and provides improved performance in almost all cases. More specifics are provided in succeeding paragraphs. Users of the Korn shell can achieve this performance improvement by aliasing as follows:

```
alias -x grep="cgrep -G"
  alias -x fgrep="cgrep -F"
```

The alias statements should be in the *.env* file (the file pointed to by the *ENV* environment variable), rather than in your *.profile.* Otherwise, these aliases are not set in subshells if you are using *ksh93* or any other version of the shell that does not support the exporting of aliases.

cgrep ranges from 30 times as fast (user time) as *grep* to 25% faster than *grep*, depending on the expressions matched, options used, and number of matches found. In a search for a five character fixed string, using no options, and with a small percentage of the lines of text containing a match, *cgrep* is about 8 times as fast as *grep*.

The performance advantage of *cgrep* over *grep* decreases slowly as the percentage of lines matched increases. In cases where a match is found on most of the lines in the text, *cgrep* typically performs form 25% faster to twice as fast (user time) as grep.

21. COMPATIBILITY

cgrep is almost completely upward-compatible with *grep* or *fgrep*. The only known exception is that for *cgrep*, *patterns* that begin with a '+' (or a '-') must be preceded by the **-e** option flag or placed in a file, while for *grep* or *fgrep*, *patterns* beginning with a '+' may be used directly. When *cgrep* is used in shell scripts, it is a good idea to precede *patterns* with **-e**.

22. LIMITATIONS

If *patterns* begin with a '-' or a '+', the **-e** option or the **-f** option must be used.

Only a **vt100** terminal type is supported.

The *cgrep* command does not allow back-referencing (available in **grep** mode) in multiple regular expressions, in regular expressions that use approximate matching, or in regular expressions that match multiple lines of text. If back-referencing is used, only a single regular expression may be specified for the pattern, delimiter, or trail pattern in which back-referencing is used, approximate matching may not be used for that regular expression, and the text that matches that regular expression must be on a single line.

Provided that sufficient memory is available, *cgrep* uses an input buffer that is large enough to hold the largest input file, (or all the input files at once if **-M** was specified). However, if the files are too big or if reading from standard input, the input buffer may not be large enough to hold all the input at once. This does not cause problems unless the size of a single window is very large. In this case, the portion of any window including the matched line and all preceding lines not overlapping a preceding window cannot contain more characters than the size of the input buffer. If a beginning delimiter is specified rather than a constant number of preceding lines, this portion also cannot contain more than 4096 lines. If either limitation is exceeded, lines are discarded from the beginning of the portion until the limitations are met.

When the **-o** option is used, the portion of any window that overlaps a preceding window cannot contain more than the size of the input buffer. If either a beginning or an ending delimiter is also specified, this portion cannot contain more than 4096 lines. If either limitation is exceeded, lines are discarded from the beginning of the window until the limitations are met. If the matching line is itself discarded, the entire window is discarded.

23. VERSION

This manual page describes Version 8.05 of *cgrep*.

CHGRP

***** See Warning in Section 1.1 *****

1. NAME

chgrp

3. DESCRIPTION

See *chown*.

CHMOD

***** See Warning in Section 1.1 *****

1. NAME

chmod - Change mode

2. SYNOPSIS

chmod mode files

3. DESCRIPTION

Chmod changes the permissions of the named *files* according to absolute or symbolic*mode*. An absolute *mode* is an octal number constructed from the OR of the following modes:

4000	Set user on execution
2000	Set group on execution
1000	Sticky bit, see *chmod(2)*
0400	Read by owner
0200	Write by owner
0100	Execute (search in directory) by owner
0070	Read, write, execute (search) by group
0007	Read, write, execute (search) by others

A symbolic *mode* has the form:
[" who "] " op permission " [" op permission "]

The *who* part is a combination of the letters **u** (for user's permissions), **g** (group) and **o** (other). The letter **a** stands for **ugo ;** the default if *who* is omitted.

Op can be **+** to add *permission* to the file mode, **-** to take away *permission*, or **=** to assign *permission* absolutely (all other bits will be reset).

Permission is any combination of the letters **r** (read), **w** (write), **x** (execute), **s** (set owner or group ID), and **t** (save text, or sticky); **u , g ,** or **o** indicate that *permission* is to be taken from the current mode. Omitting *permission* is only useful with **=** to take away all permissions.

Multiple symbolic modes separated by commas may be given. Operations are performed in the order specified. The letter **s** is only useful with **u** or **g**, and **t** only works with **u.**

Only the owner of a file (or the super user) may change its mode.

4. EXAMPLES

The first example denies write permission to others; the second makes a file executable:

```
chmod o-w file
  chmod +x file
```

5. SEE ALSO

ls(1)

CHOWN

***** See Warning in Section 1.1 *****

1. NAME

chown, chgrp - Change group or owner

2. SYNOPSIS

chown owner file ...

chgrp group file ...

3. DESCRIPTION

Chown changes the owner of the *files* to *owner*. The owner may be either a decimal user ID or a login name found in the password file.

Chgrp changes the group of the *files* to *group* . The group may be either a decimal group ID or a group name found in the group file.

4. FILES

/etc/passwd
/etc/group

CLOSEWD

***** See Warning in Section 1.1 *****

1. NAME

closewd - Close the window that is opened by *openwd*

2. SYNOPSIS

closewd

3. DESCRIPTION

Closewd closes the window opened by *openwd.* After *closewd,* processes may not open (with write) block devices for mounted file systems. This command should follow *openwd.* But its use is not mandatory since the window is closed automatically by the file manager (20 minutes after open).

4. SEE ALSO

openwd(1), fsdb(1)

CLRFS

***** See Warning in Section 1.1 *****

1. NAME

clrfs - Construct a file system

2. SYNOPSIS

clrfs special blocks [i-node blocks]

3. DESCRIPTION

Clrfs constructs a file system by writing on the special file according to the directions found in the remainder of the command line. If the second argument is given as a string of digits, *clrfs* builds a file system with a single empty directory on it.

The size of the file system is the value of *blocks* interpreted as a decimal number. The boot block is left uninitialized. If the optional number of *i-node blocks* is given, the i-list consists of eight times that value (that is, there are eight i-nodes per i-node block). If the optional number of *i-node blocks* is not given, the default is the number of *blocks* divided by four.

CLRI

***** See Warning in Section 1.1 *****

1. NAME

clri - Clear i-node

2. SYNOPSIS

/etc/clri filesystem i-numbers

3. DESCRIPTION

Clri writes zeros on the 64 bytes occupied by the i-nodes numbered **i-number**. The **filesystem** argument must be a special filename referring to a device containing a file system. After *clri*, any blocks in the affected file show up as "missing" in an *ichk* of the **filesystem**.

Read and write permission is required on the specified **filesystem** device. The i-node becomes allocatable.

The primary purpose of this routine is to remove a file which does not appear in any directory. If it is used to remove an i-node which appears in a directory, care should be taken to track down the entry and remove it. Otherwise, when the i-node is reallocated to some new file, the old entry still points to that file. At that point, removing the old entry destroys the new file. The new entry again points to an unallocated i-node; so, the whole cycle is likely to be repeated again and again.

4. SEE ALSO

ichk(1)

5. LIMITATIONS

If the file is open, *clri* is likely to be ineffective.

CMP

***** See Warning in Section 1.1 *****

1. NAME

cmp - Compare two files

2. SYNOPSIS

cmp[**-l**] [**-s**] file1 file2

3. DESCRIPTION

Cmp compares two files. (If *file1* is **-**, the standard input is used.) Under default options, *cmp* makes no comment if the files are the same; if they differ, it announces the byte and line number at which the difference occurred. If one file is an initial subsequence of the other, that fact is noted.

Options:

-l	Print the byte number (decimal) and the differing bytes (octal) for each difference.
-s	Print nothing for differing files; return codes only.

4. SEE ALSO

diff(1)

5. DIAGNOSTICS

Exit code 0 is returned for identical files, 1 for different files, and 2 for an inaccessible or missing argument.

COFLSB

***** See Warning in Section 1.1 *****

1. NAME

coflsb - Clear off-line superblock

2. SYNOPSIS

coflsb<partition anme> <mount point>

3. DESCRIPTION

Coflsb locates the given *partition name* in the off-line SG database and extracts the disk number, partition number, partition block count, and partition inode block count for that partition. This information is then used to update the superblock of the named off-line partition. The updated off-line partition is then mounted on /tmp/ofl/ *mount point.* When the partition is successfully mounted, the *coflsb* process goes to sleep for four (4) hours, allowing data transfer to take place.

4. DIAGNOSTICS

Coflsb fails with an error code if the off-line SG database cannot be attached or read, or if the requested off-line partition cannot be successfully mounted. Appropriate error messages are printed before *coflsb* exits.

5. FILES

/tmp/dev/sgdbase	-	Temporary device file for attaching to SG database
/tmp/dev/attpar	-	Temporary device file for attaching requested off-line partition
/tmp/ofl/attpar	-	Temporary mount point for off-line SG database
/tmp/ofl/<mount point>	-	Temporary mount point for requested off-line partition

6. EXAMPLES

The following command mounts the off-line no5sodd1 partition on /tmp/ofl/fubar:

```
coflsb no5sodd1 fubar
```

7. CAVEATS

Since the off-line partition is only accessible while *coflsb* is active, this command is run asynchronously (in the background). Also, *coflsb* assumes that the odd numbered disks are the off-line disks.

8. SEE ALSO

fsinit(1)

COMPRESS

***** See Warning in Section 1.1 *****

1. NAME

compress, uncompress, zcat - Compress file, uncompress file, cat compressed file

2. SYNOPSIS

compress[**-f**] [**-v**] [**-c**] [**-V**] [**-b** *bits*] [*name ...*]
uncompress[**-f**] [**-v**] [**-c**] [**-V**] [*name ...*]
zcat[**-V**] [*name ...*]

3. DESCRIPTION

Compress reduces the size of the named files using adaptive Lempel-Ziv coding. Whenever possible, each file is replaced by one with the extension, **.Z**, while keeping the same ownership modes, access and modification times. If no files are specified, the standard input is compressed to standard output. Compressed files can be restored to their original form by using *uncompress* or *zcat*.

The **-f** option forces compression of *name.* This is useful for compressing an entire directory, even if some of the files do not shrink. If **-f** is not given and *compress* is run in the foreground, the user is prompted as whether to overwritten an existing file.

The **-c** option makes *compress/uncompress* write to the standard output; no files are changed. The nondestructive behavior of *zcat* is identical to that of *uncompress* **-c**.

Compress uses the modified Lempel-Ziv algorithm popularized in "A Technique for High Performance Data Compression," Terry A. Welch, *IEEE Computer*, Vol. 17, No. 6 (June 1984), Pages 8 through 19. Common substrings in the file are first replaced by 9-bit codes 257 and up. When code 512 is reached, the algorithm switches to 10-bit codes and continues to use more bits until the limit specified by the **-b** flag is reached (default 16). *Bits* must be between 9 and 16. The default can be changed in the source to allow *compress* to be run on a smaller machine.

After the *bits* limit is attained, *compress* periodically checks the compression ratio. If it is increasing, *compress* continues to use the existing code dictionary. However, if the compression ratio decreases, *compress* discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next "block" of the file.

Note that the **-b** flag is omitted for *uncompress*, since the *bits* parameter specified during compression is encoded within the output, along with a magic number to ensure that neither decompression of random data nor recompression of compressed data is attempted.

The amount of compression obtained depends on the size of the input, the number of *bits* per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50 to 60 percent. Compression is generally much better than that achieved by Huffman coding (as used in *pack*), or adaptive Huffman coding (*compact*), and takes less time to compute.

Under the **-v** option, a message is printed yielding the percentage of reduction for each file compressed.

If the **-V** option is specified, the current version and compile options are printed on stderr.

Exit status is normally 0; if the last file is larger after (attempted) compression, the status is 2; if an error occurs, exit status is 1.

4. DIAGNOSTICS

Usage: compress [-dfvcV] [-b maxbits] [file ...] Invalid options were specified on the command line.

Missing maxbits Maxbits must follow **-b**.

file: not in compressed format The file specified to *uncompress* has not been compressed.

file: compressed with *xx* bits, can only handle *yy* bits *File* was compressed by a program that could deal with more *bits* than the compress code on this machine. Recompress the file with smaller *bits*.

File: already has .Z suffix -- no change The file is assumed to be already compressed. Rename the file and try again.

file:filename too long to tack on .Z The file cannot be compressed because its name is longer than 12 characters. Rename and try again.

file:already exists; do you wish to overwrite (y or n)? Respond "y" if you want the output file to be replaced; "n" if not.

uncompress: corrupt input A SIGSEGV violation was detected which usually means that the input file has been corrupted.

Compression: *xx.xx%* Percentage of the input saved by compression. (Relevant only for **-v**.)

-- not a regular file: unchanged When the input file is not a regular file (for example, a directory), it is left unaltered.

-- has *xx* other links: unchanged The input file has links; it is left unchanged. See *ln* (1) for more information.

-- file unchanged No savings is achieved by compression. The input remains unchanged.

To enter this command from the craft shell, compressing file "/tmp/abc", enter the following:
For MML:

EXC:ENVIR:UPROC,FN="/bin/compress",ARGS="/tmp/abc";

For PDS enter:

EXC:ENVIR:UPROC,FN"/bin/compress",ARGS"/tmp/abc"!

5. BUGS

Although compressed files are compatible between machines with large memory, **-b** 12 should be used for file transfer to architectures with a small process data space (64kb or less, as exhibited by the Intel 80286, etc.).

CP

***** See Warning in Section 1.1 *****

1. NAME

cp, ln, mv - Copy, link, or move files

2. SYNOPSIS

cp file1 [file2 ...] target

ln file1 [file2 ...] target

mv file1 [file2 ...] target

3. DESCRIPTION

File1 is copied (linked, moved) to *target*. Under no circumstance can *file1* and *target* be the same [take care when using *sh* (1) metacharacters]. If *target* is a directory, then one or more files are copied (linked, moved) to that directory.

If *mv* determines that the mode of *target* forbids writing, it prints the mode and reads the standard input for one line (if the standard input is a terminal). If the line begins with **y**, the move takes place; if not, *mv* exits.

Only *mv* allows *file1* to be a directory, in which case the directory rename occurs only if the two directories have the same parent.

4. SEE ALSO

cpio(1), rm(1)

5. BUGS

If *file1* and *target* lie on different file systems, *mv* must copy the file and delete the original. In this case, the owner name becomes that of the copying process and any linking relationship with other files is lost.

Ln does not link across file systems.

CPIO

***** See Warning in Section 1.1 *****

1. NAME

cpio - Copy file archives in and out

2. SYNOPSIS

cpio -o [acBv]
cpio -i [BcdmrtuvfsSb6] [patterns]
cpio -p [adlmruv] directory

3. DESCRIPTION

Cpio -o (copy out) reads the standard input to obtain a list of pathnames and copies those files onto the standard output together with pathname and status information.

Cpio -i (copy in) extracts files from the standard input which is assumed to be the product of a previous **cpio -o** . Only files with names that match *patterns* are selected. *Patterns* are given in the name-generating notation of *sh* (1). In *patterns* , meta-characters **?, ***, and **[...]** match the slash **/** character. Multiple *patterns* may be specified and if no *patterns* are specified, the default for *patterns* is ***** (that is, select all files). The extracted files are conditionally created and copied into the current directory tree based upon the options described in succeeding paragraphs.

Cpio -p (pass) reads the standard input to obtain a list of pathnames of files that are conditionally created and copied into the destination *directory* tree based upon the options described in the following legend.

The meanings of the available options are as follows:

a	Reset access times of input files after they have been copied.
B	Input/output is to be blocked 5,120 bytes to the record (does not apply to the *pass* option; meaningful only with data directed to or from **/dev/rmt?**).
d	*Directories* are to be created as needed.
c	Write *header* information in ASCII character form for portability.
r	Interactively *rename* files. If the user types a null line, the file is skipped.
t	Print a *table of contents* of the input. No files are created.
u	Copy *unconditionally* (normally, an older file does not replace a newer file with the same name).
v	*Verbose* : causes a list of filenames to be printed. When used with the **t** option, the table of contents looks like the output of an **ls -l** command [see *ls* (1)].
l	Whenever possible, link files rather than copying them. Usable only with the **-p** option.
m	Retain previous file modification time. This option is ineffective on directories that are being copied.
f	Copy in all files except those in *patterns*.
s	Swap bytes. Use only with the **-i** option.

S	Swap halfwords. Use only with the **-i** option.
b	Swap both bytes and halfwords. Use only with the **-i** option.
6	Process an old (that is, *UNIX*$^{®}$ System *Sixth* Edition format) file. Only useful with **-i** (copy in).

4. EXAMPLES

The following first example copies the contents of a directory into an archive; the second duplicates a directory hierarchy:

```
ls | cpio -o >/dev/mt08
  cd olddir
  find . -depth -print | cpio -pdl newdir
```

The trivial case **find . -depth -print | cpio -oB >/dev/rmt08** can be handled more efficiently by using the following command line:

```
find . -cpio /dev/mt08
```

5. SEE ALSO

find(1)

6. BUGS

Pathnames are restricted to 128 characters. If there are too many unique linked files, the program runs out of memory, and, thereafter, linking information is lost. Only the super user can copy special files. The **-B** option does not work with certain magnetic tape drives.

CRON

***** See Warning in Section 1.1 *****

1. NAME

cron - Clock daemon

2. SYNOPSIS

/etc/cron

3. DESCRIPTION

Cron executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in crontab files; users can submit their own crontab file via the *crontab* command. Commands which are to be executed only once may be submitted via the **at** command. Since **cron** never exits, it should only be executed once.

Cron only examines crontab files and **at** command files during process initialization and when a file changes. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

3.1 Queue Definitions

Cron can limit dynamically the number of concurrently running jobs. **cron** can also maintain up to 26 separate queues and control the number of jobs executed in each one. The file **queuedefs** is used to maintain definitions for all queues. Each line of the file must have the following format:

*q.***NNj***NN***n***NN***w**

Where: *q* **is a letter** *a-z* **indicating the job queue.**

*NN***j** is the limit on jobs that can be running at any one time for the job queue. (*NN* is an integer; its default is 100.)

*NN***n** is the **nice (1)** value that is assigned each command executed for the job queue. (Default is 2.)

*NN***w** is the time (in seconds) to wait before retrying to execute a command if all the criteria for running the command are not met. (Default is 60.)

Empty fields are initialized to the default values. The following is an example of a **queuedefs** file.

```
 a.4j1n

 b.2j2n90w

 c.0n10j

 n.120w4n1j
```

Changes to queue definitions take effect before the next job is executed by the **cron** daemon. If this file does not exist, the default values are used.

3.2 Prototype File

The prototype file provides a method to customize **at** command files by controlling what information is written into the **at** job file. If a file named **.proto.q** exists (where **q** indicates a queue name), this file is copied into the job file. Otherwise, the file **.proto** is used.

The following substitutions are made during creation of an **at** job file.

$m	Current file creation mask of the user.
$l	Current file size limit (not implemented)
$d	Name of the current working directory
$t	Time (in seconds) that the jobs is scheduled to execute
$<	Read standard input until EOF is reached.

The following is an example of a prototype file:

```
 cd $d

 ulimit $l

 umask $m

 $<
```

The **at** command exits with an error if no prototype file exists.

3.3 Log Information

Cron logs all command invocations, terminations, and status information in the file **log** . Records that begin with the character **>** pertain to command invocations. Two invocation records are written for each command execution. The first displays the command that is being executed; the second contains the login name, process id, job queue, and a timestamp for when the command was invoked. Command termination records begin with the character **<** and are similar to the second invocation record, except that a nonzero termination status or exit status is also printed. Records that begin with an **!** indicate status information.

3.4 Files to Limit Access to the Facility

There is potential for misuse of system resources through use of the **crontab** and **at** commands. Cron provides a method to restrict user access to it. The files **cron.allow** and **at.allow** contain login names of users (one per line) allowed access to the **crontab** and **at** commands, while the files **cron.deny** and **at.deny** contain login names of users denied access to the commands.

When a user submits a crontab file, **crontab** checks **cron.allow** for a list of users permitted to have a crontab. If that file does not exist, the file **cron.deny** is scanned for users who are denied crontabs. If neither file exists, only root is allowed to have a crontab file. The same scheme is used for determining access to the **at** command. The null file **cron.allow** would mean no user is allowed a crontab while a null file **cron.deny** would mean no user is denied a crontab.

3.5 Defining Queues

The **at** command can queue jobs in one of 26 different queues, with the **cron** daemon controlling the number of executions for each queue. This mechanism can be used by system administrators to limit the number of simultaneous executions of high-resource commands. Running the **at** command with **-q** *c* as the first argument queues the command in queue *c* . The default queue is **a** . A special queue **b** is defined to be a batch queue; jobs in this queue run whenever the defined maximum level is not exceeded (specified in the **queuedefs** file). Cron executions are limited by the definition of the queue **c**. Jobs in all other queues run at the time specified on the command line.

The ability to define queues gives administrators a way to restrict executions for commands. An example of its use would be in providing a batch **sort (1)** facility. A queue needs defining in the **queuedefs** file to spool **sort** commands, create a prototype file (if necessary), and replace the **sort** command with a shell that creates an **at** job and invokes the real **sort** . The real **sort** command should be moved out of a standard bin location.

4. FILES

/unixa/lib/cron	Main cron directory
/unixa/lib/cron/log	Accounting information
/unixa/spool/cron	Spool area

5. SEE ALSO

at(1), crontab(1), sh(1)

6. DIAGNOSTICS

A history of all actions taken by cron are recorded in **/unixa/lib/cron/log.**

CRONTAB

***** See Warning in Section 1.1 *****

1. NAME

crontab - User crontab file

2. SYNOPSIS

crontab [file]

crontab -r

crontab -l

3. DESCRIPTION

Crontab copies the specified file or standard input, if no file is specified, into a directory that holds all crontabs of users. The -r option removes a crontab of a user from the crontab directory. *Crontab* -l lists the crontab file for the invoking user.

A user is permitted to use *crontab* if their name appears in the file **/unixa/lib/cron/cron.allow.** If that file does not exist, the file **/unixa/lib/cron/cron.deny** is checked to determine if the user should be denied access to *crontab*. If neither file exists, only root is allowed to submit a job. If either file is **at.deny**, global usage is permitted. The allow/deny files consist of one user name per line.

A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns that specify the following:

minute (0-59),

hour (0-23),

day of the month (1-31),

month of the year (1-12),

day of the week (0-6 with 0=Sunday).

Each of these patterns may be either an asterisk (meaning all legal values) or a list of elements separated by commas. An element is either a number or two numbers separated by a minus sign (meaning an inclusive range). Note that the specification of days may be made by two fields (day of the month and day of the week). Both are adhered if both are specified as a list of elements. For example, 0 0 1,15 * 1 would run a command on the first and fifteenth of each month, as well as on every Monday. To specify days by only one field, the other field should be set to * (for example, 0 0 * * 1 would run a command only on Mondays).

The sixth field of a line in a crontab file is a string that is executed by the shell at the specified times. A percent character in this field (unless escaped by \) is translated to a new-line character. Only the first line (up to a % or end of line) of the command field is executed by the shell. The other lines are made available to the command as standard input.

The shell is invoked from your **$HOME** directory with an **arg0** of **sh**. Users who desire to have their *.profile* executed must explicitly do so in the crontab file. *Cron* supplies a default environment for every shell, defining **HOME, LOGNAME, SHELL(=/bin/sh)**, and **PATH(=/bin:/usr/bin::)**.

NOTE: Users should remember to redirect the standard output and standard error of their commands! If this is not done, any generated output or error is mailed to the user.

4. FILES

/unixa/lib/cron	# Main cron directory
/unixa/spool/cron/crontabs	# Spool area
/unixa/lib/cron/log	# Accounting information
/unixa/lib/cron/cron.allow	# List of allowed users
/unixa/lib/cron/cron.deny	# List of denied users

5. SEE ALSO

cron(1), sh(1)

CUT

***** See Warning in Section 1.1 *****

1. NAME

cut - Cut out selected fields or each line of a file

2. SYNOPSIS

cut-clist [file1 file2 ...]

cut-flist [-**d** char] [-**s**] [file1 file2 ...]

3. DESCRIPTION

Cut cuts out columns from a table or fields from each line of a file; in database parlance, it implements the projection of a relation. The fields as specified by *list* can be fixed length, that is, character positions as on a punched card (**-c** option) or the length can vary from line to line and be marked with a field delimiter character like *tab* (**-f** option). *Cut* can be used as a filter; if no files are given, the standard input is used.

The meanings of the options are as follows:

list	A comma-separated list of integer field numbers (in increasing order), with optional **-** to indicate ranges as in the **-o** option of *nroff / troff* for page ranges; for example, *1,4,7 ; 1-3,8 ; -5,10* (short for **1-5,10**); or *3-* (short for third through last field).
-c*list*	The *list* following *-c* (no space) specifies character positions (for example, *-c1-72* would pass the first 72 characters of each line).
-f*list*	The *list* following *-f* is a list of fields assumed to be separated in the file by a delimiter character (see **-d**); for example, **-f1,7** copies the first and seventh field only. Lines with no field delimiters are passed through intact (useful for table subheadings), unless **-s** is specified.
-d*char*	The character following **-d** is the field delimiter (**-f** option only). Default is *tab.* Space or other characters with special meaning to the shell must be quoted.
-s	Suppresses lines with no delimiter characters in case of **-f** option. Unless specified, lines with no delimiters are passed through untouched.

Either the **-c** or **-f** option must be specified.

4. HINTS

Use *grep (1)* to make horizontal "cuts" (by context) through a file, or *paste (1)* to put files together column-wise (that is, horizontally). To reorder columns in a table, use *cut* and *paste.*

5. EXAMPLES

```
cut -d: -f1,5 /etc/passwd
  (Mapping of user IDs to names)
name=`;who am i | cut -f1 -d" "`
  (To set name to current login name).
```

6. DIAGNOSTICS

line too long A line can have no more than 1023 characters or fields.

bad list for c/f option Missing **-c** or **-f** option or incorrectly specified *list.* No error occurs if a line has fewer fields than the *list*requires.

no fields The *list* is empty.

7. SEE ALSO

grep(1), paste(1)

CX

***** See Warning in Section 1.1 *****

1. NAME

cx - 3B20D computer core file examiners

2. SYNOPSIS

cx [-**!aphx**] [-**v** vaddr [-**n** nbytes]] ... **file**

3. DESCRIPTION

Cx prints the segment images of the specified file. *cx* resides on the 3B20D computer. When invoked without options, *cx* prints all data from all nonexecutable segments in the given file. Options to modify this behavior follow:

-!	Prints descriptions of the meaning of each flag. Illegal command lines generate a terse summary of usage without the descriptions.
-a	Prints data for all nonexecutable segment images. This is the default action when no **-v, -h,** or **-x** flags are present.
-h	Prints a header for every segment. This flag also suppresses the printing of segment data, but **-a, -v,** or **-x** causes the designated data to be printed.
-p	Requests the *pfile* meanings, which are different in a small number of cases. Each segment's header contains a set of attribute flags. *cx* normally decodes the flags using their *execution* time meanings.
-x	Prints data for all executable segments. Without **-x**, the data are suppressed.
-v *vaddr*	Starts printing data from the given address in the segment(s) containing virtual address *vaddr*. Unless **-n** is also given, the rest of the segment is printed. Using one or more **-v** specifications causes other segment images not to be printed, but **-a** and **-x** may be used as an override.
n *nbytes*	Prints *nbytes* bytes instead of printing all data of a segment. A count of zero means print the remainder of the image. This option applies only to the preceding **-v** , which must be present.

Values for *addr* and *nbytes* are interpreted as hexadecimal numbers.

A format of a core file must be the same as that written by the process manager, which is very similar to executable *pfiles* .

Usually the process manager places core files in */cdmp/name*; where *name* is the ASCII name in the PCB.

In actual core files, the segment images occur in the same order as they appeared in the segment list of the process. The PCB is first, and the stack is second. Those segment images in *pfiles* are empty.

More than one segment can be mapped to the same virtual address, but only one can be active. **-v** directives apply to all mapped segments, not just the active one.

DATE

***** See Warning in Section 1.1 *****

1. NAME

date - Print and set the date

2. SYNOPSIS

date [mmddhhmmyy] [+format]

3. DESCRIPTION

If no argument is given or if the argument begins with **+**, the current date and time are printed. Otherwise, the current date is set. The first *mm* is the month number; *dd* is the day number in the month; *hh* is the hour number (24-hour system); the second *mm* is the minute number; *yy* is the last 2 digits of the year number. The following example sets the date to Oct 8, 12:45 AM 1985:

```
date 1008004585
```

The system operates in GMT. *Date* takes care of the conversion to and from local standard and daylight time.

If the argument begins with **+**, the output of *date* is under the control of the user. All output fields are of fixed size (zero padded if necessary). Each field descriptor is preceded by **%** and is replaced in the output by its corresponding value. A single **%** is encoded by **%%**. All other characters are copied to the output without change. The string is always terminated with a new-line character.

Field Descriptors:

n	Insert a new-line character
t	Insert a tab character
m	Month of year - 01 to 12
d	Day of month - 01 to 31
y	Last 2 digits of year - 00 to 99
D	Date as mm/dd/yy
H	Hour - 00 to 23
M	Minute - 00 to 59
S	Second - 00 to 59
T	Time as HH:MM:SS
j	Day of year - 001 to 366
w	Day of week - Sunday = 0
a	Abbreviated weekday - Sun to Sat
h	Abbreviated month - Jan to Dec
r	Time in AM/PM notation

4. EXAMPLE

```
date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'
```

Generates the following output:

```
DATE: 08/01/76
  TIME: 14:45:05
```

5. DIAGNOSTICS

No permission If the user is not the super user and tries to change the date;

Bad conversion If the date set is syntactically incorrect;

Bad format character If the field descriptor is not recognizable.

6. FILES

/dev/kmem

7. WARNING

If the system is call processing, use the PDS/MML command **SET:CLK** to change the date.

DC

***** See Warning in Section 1.1 *****

1. NAME

dc - Desk calculator

2. SYNOPSIS

dc[file]

3. DESCRIPTION

Dc is an arbitrary precision arithmetic package. Ordinarily it operates on decimal integers, but one may specify an input base, output base, and a number of fractional digits to be maintained. The overall structure of *dc* is a stacking (reverse polish) calculator. If an argument is given, input is taken from that file until its end, then from the standard input. The following constructions are recognized:

number	The value of the number is pushed on the stack. A number is an unbroken string of the digits 0-9. It may be preceded by an underscore (_) to input a negative number. Numbers may contain decimal points.
+ - / * % ^	The top two values on the stack are added (+), subtracted (-), multiplied (*****), divided (**/**), remaindered (**%**), or exponentiated (**^**). The two entries are popped off the stack; the result is pushed on the stack in their place. Any fractional part of an exponent is ignored.
s*x*	The top of the stack is popped and stored into a register named *x* , where *x* may be any character. If the **s** is capitalized, *x* is treated as a stack and the value is pushed on it.
l*x*	The value in register *x* is pushed on the stack. The register *x* is not altered. All registers start with zero value. If the **l** is capitalized, register *x* is treated as a stack and its top value is popped onto the main stack.
d	The top value on the stack is duplicated.
p	The top value on the stack is printed. The top value remains unchanged. **P** interprets the top of the stack as an ASCII string, removes it, and prints it.
f	All values on the stack are printed.
q	Exits the program. If executing a string, the recursion level is popped by two. If **q** is capitalized, the top value on the stack is popped and the string execution level is popped by that value.
x	Treats the top element of the stack as a character string and executes it as a string of *dc* commands.
X	Replaces the number on the top of the stack with its scale factor.
[...]	Puts the bracketed ASCII string onto the top of the stack.
< x > x = x	The top two elements of the stack are popped and compared. Register *x* is evaluated if they obey the stated relation.
v	Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.
!	Interprets the rest of the line as a UTS system command.

c	All values on the stack are popped.
i	The top value on the stack is popped and used as the number radix for further input.
I	Pushes the input base on the top of the stack.
o	The top value on the stack is popped and used as the number radix for further output.
O	Pushes the output base on the top of the stack.
k	The top of the stack is popped, and that value is used as a non-negative scale factor: the appropriate number of places are printed on output, and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base is reasonable if all are changed together.
z	The stack level is pushed onto the stack.
Z	Replaces the number on the top of the stack with its length.
?	A line of input is taken from the input source (usually the terminal) and executed.
; :	Are used for array operations.

4. EXAMPLE

This example prints the first ten values of n!:

```
[la1+dsa*pla10>y]sy
 0sa1
 lyx
```

5. DIAGNOSTICS

x is unimplemented
x is an octal number.

Stack empty
Not enough elements on the stack to do what was asked.

Out of space
The free list is exhausted (too many digits).

Out of headers
Too many numbers being kept around.

Out of pushdown
Too many items on the stack.

Nesting Depth
Too many levels of nested execution.

DD

***** See Warning in Section 1.1 *****

1. NAME

dd - Convert and copy a file

2. SYNOPSIS

dd[option=value] ...

3. DESCRIPTION

Dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block size may be specified to take advantage of raw physical I/O.

option	*values*
if= file	Input filename; standard input is default
of= file	Output filename; standard output is default
ibs= n	Input block size *n* bytes (default 512)
obs= n	Output block size (default 512)
bs= n	Set both input and output block size, superseding *ibs* and *obs*; also, if no conversion is specified, it is particularly efficient since no in-core copy need be done
cbs= n	Conversion buffer size
skip= n	Skip *n* "" input records before starting copy
seek= n	Seek *n* records from beginning of output file before copying
count= n	Copy only *n* "" input records
conv=**ascii**	Convert EBCDIC to ASCII
ebcdic	Convert ASCII to EBCDIC
ibm	Slightly different map of ASCII to EBCDIC
lcase	Map alphabetics to lowercase
ucase	Map alphabetics to uppercase
swab	Swap every pair of bytes
noerror	Do not stop processing on an erroc
sync	Pad every input record to *ibs*
... , ...	Several comma-separated conversions

Where sizes are specified, a number of bytes is expected. A number may end with **k, b**, or **w** to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by **x** to indicate a product.

Cbs is used only if *ascii* or *ebcdic* conversion is specified. In the former case, *cbs* characters are placed into the conversion buffer, converted to ASCII, and trailing blanks trimmed and new-line added before sending the line to the output. In the latter case, ASCII characters are read into the conversion buffer, converted to EBCDIC, and blanks added to make up an output record of size *cbs* .

After completion, *dd* reports the number of whole and partial input and output blocks.

4. EXAMPLE

This command reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per record into the ASCII file **x**:

```
dd if=/dev/mt00 of=x ibs=800 cbs=80 conv=ascii,lcase
```

Note the use of raw magtape. *Dd* is especially suited to I/O on the raw physical devices because it allows reading and writing in arbitrary record sizes.

5. SEE ALSO

cp(1)

6. DIAGNOSTICS

f+p records in(out)	Numbers of full and partial records read(written)

7. BUGS

The ASCII/EBCDIC conversion tables are taken from the 256-character standard in the CACM (Communications of the Association for Computing Machinery) Nov, 1968. The *ibm* conversion corresponds better to certain IBM print train conventions. There is no universal solution.

New lines are inserted only on conversion to ASCII; padding is done only on conversion to EBCDIC. These should be separate options.

The magnetic tape (MT) unit driver program of the 3B20D has a bug in it that does not permit files of certain sizes to be written. Specifically, if the file size is (521*N)+1 (that is, 1, 513, 1025 ..., the driver fails to write out the last byte of the file. This may or may not result in an error message.

This most commonly occurs when a file is being copied to tape by use of the "dd" command as part of a non-official Lucent procedure.

This problem can be avoided by using the following command to copy files to tape:

```
COPY:TAPE:OUT
```

DF

***** See Warning in Section 1.1 *****

1. NAME

df - Report number of free disk blocks

2. SYNOPSIS

df [file-systems]

3. DESCRIPTION

Df prints out the number of free blocks available for on-line file systems by examining the counts kept in the in-core super-blocks; *file systems* may be specified either by device name (for example, **/dev/bwm)** or by mounted directory name (for example, **/etc/bwm).** If the *file systems* argument is unspecified, the free space on all of the mounted file systems is printed. Blocks are reported as 512 byte blocks.

DGNNM

***** See Warning in Section 1.1 *****

1. NAME

dgnnm - Assign special diagnostic filename

2. SYNOPSIS

dgnnm unit-name unit-number

3. DESCRIPTION

Dgnnm creates and returns a block special device filename that it has assigned to the unit specified in the command line.

The unit has been reserved for diagnostics when this program completes. After use, it must be released by *udgnnm*.

4. FILES

/dev/ecd

/dgn/xxx

/temp/ecdxxxxxx

/tmp/xxx

5. SEE ALSO

udgnnm(1)

6. DIAGNOSTICS

Error numbers are returned. These numbers may be interpreted in *MOVEerrcod.h*.

DIFF

***** See Warning in Section 1.1 *****

1. NAME

diff - Differential file comparator

2. SYNOPSIS

diff [-**efbh**] file1 file2

3. DESCRIPTION

Diff tells what lines must be changed in two files to bring them into agreement. If *file1* (*file2*) is **-**, the standard input is used. If *file1* (*file2*) is a directory, then a file in that directory with the name *file2* (*file1*) is used. The normal output contains lines of these forms:

n1 **a** *n3,n4*
n1,n2 **d** *n3*
n1,n2 **c** *n3,n4*

These lines resemble *ed* commands to convert *file1* into *file2*. The numbers after the letters pertain to *file2*. In fact, by exchanging **a** for **d** and reading backward one may ascertain equally how to convert *file2* into *file1*. As in *ed*, identical pairs, where *n1* = *n2* or *n3* = *n4*, are abbreviated as a single number.

Following each of these lines come all the lines that are affected in the first file flagged by **<**; then all the lines that are affected in the second file flagged by **>**.

The **-b** option causes trailing blanks (spaces and tabs) to be ignored and other strings of blanks to compare equal.

The **-e** option produces a script of *a*, *c*, and *d* commands for the editor *ed*, which recreates *file2* from *file1*. The **-f** option produces a similar script, not useful with *ed*, in the opposite order. In connection with **-e**, the following shell program may help maintain multiple versions of a file. Only an ancestral file ($1) and a chain of version-to-version *ed* scripts ($2,$3,...) made by *diff* need be on hand. A "latest version" appears on the standard output.

```
(shift; cat $*; echo ´1,$p´) | ed - $1
```

Except in rare circumstances, *diff* finds a smallest sufficient set of file differences.

Option **-h** does a fast, half-hearted job. It works only when changed stretches are short and well separated, but does work on files of unlimited length. Options **-e** and **-f** are unavailable with **-h**.

4. FILES

/tmp/d?????
/usr/lib/diffh for **-h**

5. SEE ALSO

cmp(1), ed(1)

6. DIAGNOSTICS

Exit status is 0 for no differences, 1 for some differences, 2 for trouble.

7. BUGS

Editing scripts produced under the **-e** or **-f** option are naive about creating lines consisting of a single period (**.**).

DIRNAME

***** See Warning in Section 1.1 *****

1. NAME

dirname

3. DESCRIPTION

See *basename*.

DLSUM

***** See Warning in Section 1.1 *****

1. NAME

dlsum - "Date-less" sum and block counts of a file

2. SYNOPSIS

dlsum [-**r**] file

3. DESCRIPTION

Dlsum calculates and prints a 16-bit checksum of the named file excluding the time-date stamp and header version information. A file block count is also printed. *Dlsum* is typically used to check the "object" content of a file regardless of header and version time stamping.

4. SEE ALSO

sum(1)

5. DIAGNOSTICS

Read error is indistinguishable from end-of-file on most devices; check the block counts.

DU

***** See Warning in Section 1.1 *****

1. NAME

du - Summarize disk usage

2. SYNOPSIS

du [-**ars**] [names]

3. DESCRIPTION

Du gives the number of blocks contained in all files and (recursively) directories within each directory and file specified by the *names* argument. The block count includes the indirect blocks of the file. If the argument *names* is missing, **q** is used.

The optional argument **-s** causes only the grand total (for each of the specified *names*) to be given. The optional argument **-a** causes an entry to be generated for each file. Absence of either causes an entry to be generated for each directory only.

Du is normally silent about directories that cannot be read, files that cannot be opened, etc. The **-r** option causes *du* to generate messages in such instances.

A file with two or more links is only counted once.

4. BUGS

If the **-a** option is not used, nondirectories given as arguments are not listed.

If there are too many distinct linked files, *du* counts the excess files more than once.

Files with holes get an incorrect block count.

ECHO

***** See Warning in Section 1.1 *****

1. NAME

echo - Echo arguments

2. SYNOPSIS

echo [arg] ...

3. DESCRIPTION

Echo writes arguments separated by blanks and terminated by a new-line on the standard output. It also understands C-like escape conventions; beware of conflicts with the shell's use of \\:

\b	Backspace
\c	Print line without new-line
\f	Form-feed
\n	New-line
\r	Carriage return
\t	Tab
\\	Backslash
\n	The 8-bit character whose ASCII code is the 1-, 2-, or 3-digit octal number *n*, which must start with a zero.

Echo is useful for producing diagnostics in command files and for sending known data into a pipe.

4. SEE

sh(1)

ED

***** See Warning in Section 1.1 *****

1. NAME

ed, red - Text editor

2. SYNOPSIS

ed[-] [file]
red[-] [file]

3. DESCRIPTION

Ed is the standard text editor. If the *file* argument is given, *ed* simulates an *e* command on the named file; that is, the file is read into the buffer of *ed* so that it can be edited. The optional "**-**" suppresses the printing of character counts by *e* , *r* , and *w* commands; diagnostics from *e* and *q* commands; and of the **!** prompt after a **!** *shell command*. *Ed* edits a copy of the file to be edited. A copy, of the files, resides in a temporary file called the *buffer* . There is only one buffer. Changes made to the copy have no effect on the file until a *w* (write) command is given.

Red is a restricted version of *ed*. The *ed* edits only files in the current directory and prohibits the execution of shell commands via **!** *shell command*. Attempts to bypass these restrictions result in an error message (*restricted shell*).

Both *ed* and *red* support a formatting capability. After including a format specification as the first line of *file* and invoking *ed* with your terminal in **"stty -tabs** or **"stty tab3"** mode [see *stty* (1)], the specified tab stops are automatically used when scanning a *file*. The following command sets tab stops at columns 5, 10, and 15, with a maximum line length of 72 is imposed:

```
<:t5,10,15 s72:>
```

Note: While inputting text, tab characters when typed are expanded to every eighth column as is the default.

Commands to *ed* have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command*, and possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so addresses can very often be omitted.

In general, only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While *ed* is accepting text, it is in the *input mode*. In this mode, *no* commands are recognized; all input is merely collected. Input mode is exited by typing a period ("**.**") alone at the beginning of a line.

Ed supports a limited form of *regular expression* notation; regular expressions are used in addresses to specify lines and in some commands (for example, *s*) to specify portions of a line that are to be substituted. A regular expression (RE) specifies a set of character strings. A member of this set of strings is said to be *matched* by the RE.

The REs allowed by *ed* are constructed as follows:

The following *1-character RE*s match a *single* character:

1.1 An ordinary character (*not* one of those discussed in 1.2) is a 1-character RE that matches itself.

1.2 A backslash (\) followed by any special character is a 1-character RE that matches the special character itself. The special characters are as follows:

a. "**.**", *****, **[**, and \ (period, asterisk, left-square bracket, and backslash, respectively) which are always special, *except* when they appear within square brackets (**[ ]**; see 1.4).

b. ^ (caret or circumflex) which is special at the *beginning* of an *entire* RE (see 3.1 and 3.2), or when it immediately follows the left of a pair of square brackets (**[ ]**) (see 1.4).

c. **$** (currency symbol) which is special at the *end* of an entire RE (see 3.2).

d. The character used to bound (that is, delimit) an entire RE, which is special for that RE [for example, see how slash (***/***) is used in the *g* command].

1.3 A period ("**.**") is a 1-character RE that matches any character except new line.

1.4 A nonempty string of characters enclosed in square brackets (**[ ]**) is a 1-character RE that matches *any one* character in that string. If, however, the first character of the string is a circumflex (^), the 1-character RE matches any character *except* new line and the remaining characters in the string. The ^ has this special meaning *only* if it occurs first in the string. The minus (**-**) may be used to indicate a range of consecutive ASCII characters; for example, **[0-9]** is equivalent to **[0123456789]** . The **-** loses this special meaning if it occurs first (after an initial , if any) or last in the string. The right-square bracket (**]**) does not terminate such a string when it is the first character within it (after an initial , if any); for example, **[ ]a-f]** matches either a right-square bracket (**]**) or one of the letters **a** through **f** inclusive. The four characters listed in "1.2.a" are the resulting characters within such a string of characters.

The following rules may be used to construct *RE*s from 1-character REs:

2.1 A 1-character RE is an RE that matches whatever the 1-character RE matches.

2.2 A 1-character RE followed by an asterisk (*****) is an RE that matches *zero* or more occurrences of the 1-character RE. If there is any choice, the longest leftmost string that permits a match is chosen.

2.3 A 1-character RE followed by **\\ {** *m* **\\ }** , **\\ {** *m,***\\ }** , or **\\ {** *m,n* **\\ }** is an RE that matches a *range* of occurrences of the 1-character RE. The values of *m* and *n* must be nonnegative integers less than 256; **\\{** *m* **\\ }** matches *exactly m* occurrences; **\\ {** *m,***\\ }** matches *at least m* occurrences; **\\ {** *m,n* **\\ }** matches *any number* of occurrences *between m* and *n*, inclusive.

Whenever a choice exists, the RE matches as many occurrences as possible.

2.4 The concatenation of REs is an RE that matches the concatenation of the strings matched by each component of the RE.

2.5 An RE enclosed between the character sequences **\\ (** and **\\ )** is an RE that matches whatever the unadorned RE matches.

2.6 The expression *n* matches the same string of characters as was matched by an expression enclosed between **\\(** and **\\)** *earlier* in the same RE. Here *n* is a digit; the subexpression specified is that beginning with the *n*th occurrence of **\\(** counting from the left. For example, the expression **^ (. *)\\1 $** matches a line consisting of two repeated appearances of the same string.

Finally, an *entire RE* may be constrained to match only an initial segment or final segment of a line or both:

3.1 A circumflex (^) at the beginning of an entire RE constrains that RE to match an *initial* segment of a line.

3.2 A currency symbol (**$**) at the end of an entire RE constrains that RE to match a *final* segment of a line.

The construction ^*entire\f RE***$** constrains the entire RE to match the entire line.

The null RE (for example, ***//***) is equivalent to the last RE encountered. Also see the last paragraph before *FILES*.

To understand addressing in *ed* it is necessary to know that at any time there is a *current line* . Generally, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command. *Addresses* are constructed as follows:

(1) The character "**.**" addresses the current line.

(2) The character **$** addresses the last line of the buffer.

(3) A decimal number *n* addresses the *n*th line of the buffer.

(4) *´x* addresses the line marked with the mark name character *x* , which must be a lowercase letter. Lines are marked with the *k* command.

(5) An RE enclosed by slashes (*/*) addresses the first line found by searching *forward* from the line *following* the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched. Also, read the last paragraph before *FILES*.

(6) An RE enclosed in question marks (**?**) addresses the first line found by searching *backward* from the line *preceding* the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. Also, read the last paragraph before *FILES*.

(7) An address followed by a plus sign (**+**) or a minus sign (**-**) followed by a decimal number specifies that address plus (respectively minus) the indicated number of lines. The plus sign may be omitted.

(8) If an address begins with **+** or **-**, the addition or subtraction is taken with respect to the current line; for example, **-5** is understood to mean "**.-5**."

(9) If an address ends with **+** or **-**, then 1 is added to or subtracted from the address, respectively. As a consequence of this rule and of rule 8, the address **-** refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the character in addresses is entirely equivalent to **-**.) Moreover, trailing **+** and **-** characters have a cumulative effect; so, **--** refers to the current line less 2.

(10) For convenience, a comma (**,**) stands for the address pair **1,$**, while a semicolon (**;**) stands for the pair " **.,$**."

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (**,**) but addresses are also separated by a semicolon (**;**). In the latter case, the current line ("**.**") is set to the first address, and only then is the second address calculated. This feature can be used to determine the starting line for forward and backward searches (see rules 5. and 6.). The second address of any 2-address sequence must correspond to a line that follows, in the buffer, the line corresponding to the first address.

In the following list of *ed* commands, the default addresses are shown in parentheses. The parentheses are *not* part of the address but show that the given addresses are the default.

It is generally illegal for more than one command to appear on a line. However, any command (except **e, f, r, or w**) may be suffixed by **l**, **n**, or **p**. In which case, the current line is either listed, numbered, or printed, respectively, as discussed under the **l, n, and p** commands.

(.)a <text> .

The **a**ppend command reads the given text and appends it after the addressed line; " **.**" is left at the last inserted line, or, if there were none, at the addressed line. Address 0 is legal for this command; it causes the ``appended'' text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.)c <text> .

The **c**hange command deletes the addressed lines and then accepts input text that replaces these lines. The "**.**" is left at the last line input; or, if there were none, at the first line that was not deleted.

(.,.)d

The **d**elete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line.

e *file*

The **e**dit command causes the entire contents of the buffer to be deleted and the named file to be read in. The "**.**" is set to the last line of the buffer. If no filename is given, the currently remembered filename, if any, is used (see the **f** command). The number of characters read is typed; *file* is remembered for possible use as a default filename in subsequent **e**, **r**, and **w** commands. If *file* is replaced by **!**, the rest of the line is taken to be a shell [*sh*(1)] command whose output is to be read. Such a shell command is *not* remembered as the current filename. See *DIAGNOSTICS* **.**

E *file*

The **E**dit command is like **e**, except that the editor does not check to see if any changes have been made to the buffer since the last **w** command.

f *file*

If *file* is given, the **f**ilename command changes the currently remembered filename to *file*; otherwise, it prints the currently remembered filename.

(1,$)g/*RE***/***command list*

In the **g**lobal command, the first step is to mark every line that matches the given RE. Then, for every such line, the given *command list* is executed with "**.**" initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. All lines of a multiline list except the last line must be ended with a **e**; **a**, **i**, and **c** commands and associated input are permitted; the " **.**" terminating input mode may be omitted if it would be the last line of the *command list*. An empty *command list* is equivalent to the **p** command. The **g , G, v**, and **V** commands are *not* permitted in the *command list*. See also *BUGS* and the last paragraph before *FILES***.**

(1,$)G/*RE***/**

In the interactive **G**lobal command, the first step is to mark every line that matches the given RE. Every line that matches the given RE is printed, then the "**.**" is changed to that line, and anyone command (other than one of the**a**, **c**, **i**, **g**, **G** , **v**, and **V** commands) may be input and is executed.

After the execution of that command, the next marked line is printed, and so on; a new line acts as a null command; an **&** causes the most recent command to be executed again within the current invocation of **G**. Note that the commands input as part of the execution of the **G** command may address and affect *any* lines in the buffer. The **G** command can be terminated by an interrupt signal (ASCII DEL or BREAK).

h

The **h**elp command gives a short error message that explains the reason for the most recent **?** diagnostic.

H

The **H**elp command causes *ed* to enter a mode in which error messages are printed for all subsequent **?** diagnostics and also explains the previous **?** if there was one. The **H** command alternately turns this mode on and off; it is initially off.

(.)i <text> .

The **i**nsert command inserts the given text before the addressed line; "**.**" is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the **a** command only in the placement of the input text. Address 0 is not legal for this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.,.+1)j

The **j**oin command joins contiguous lines by removing the appropriate new-line characters. If exactly one address is given, this command does nothing.

(.)k*x*

The mar**k** command marks the addressed line with name *x* , which must be a lowercase letter. The address *´x* then addresses this line; "**.**" is unchanged.

(.,.)l

The **l**ist command prints the addressed lines in an unambiguous way: a few nonprinting characters (for example, *tab, backspace*) are represented by (hopefully) mnemonic overstrikes; all other non-printing characters are printed in octal, and long lines are folded. An **l** command may be appended to any other command other than **e**, **f**, **r** , or **w**.

(.,.)m*a*

The **m**ove command repositions the addressed line(s) after the line addressed by **a**. Address 0 is legal for **a** and causes the addressed line(s) to be moved to the beginning of the file; it is an error if address **a** falls within the range of moved lines; "**.**" is left at the last line moved.

(.,.)n

The **n**umber command prints the addressed lines, preceding each line by its line number and a tab character; "**.**" is left at the last line printed. The **n** command may be appended to any other command other than **e**, **f**, **r**, or **w**.

(.,.)p

The **p**rint command prints the addressed lines; "." is left at the last line printed. The **p** command may be appended to any other command other than **e**, **f**, **r**, or **w**; for example, **dp** deletes the current line and prints the new current line.

P

The editor prompts with a* for all subsequent commands. The **P** command alternately turns this mode on and off; it is initially off.

q

The **q**uit command causes *ed* to exit. No automatic write of a file is done (see *DIAGNOSTICS*).

Q

The editor exits without checking if changes have been made in the buffer since the last **w** command.

($)r *file*

The **r**ead command reads in the given file after the addressed line. If no filename is given, the currently remembered filename, if any, is used (see **e** and **f** commands). The currently remembered filename is *not* changed unless *file* is the very first filename mentioned since*ed* was invoked. Address 0 is legal for **r** and causes the file to be read at the beginning of the buffer. If the read is successful, the number of characters read is typed; the "**.**" is set to the last line read in. If *file* is replaced by **!**, the rest of the line is taken to be a shell [*sh*(1)] command whose output is to be read. For example, "$r !ls" appends current directory to the end of the file being edited. Such a shell command is *not* remembered as the current filename.

(.,.)s/*RE/replacement/* or **(.,.)s/***RE/replacement/***g**

The **s**ubstitute command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by the *replacement* if the global replacement indicator **g** appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. It is an error for the substitution to fail on *all* addressed lines. Any character, other than space or new line, may be used instead of **/** to delimit the RE and the *replacement*; "**.**" is left at the last line on which a substitution occurred. See also the last paragraph before *FILES*.

An ampersand (**&**) appearing in the *replacement* is replaced by the string matching the RE on the current line. The special meaning of **&** in this context may be suppressed by preceding it by \. As a more general feature, the characters *n*, where *n* is a digit, are replaced by the text matched by the *n* the regular subexpression of the specified RE enclosed between **\(** and **\)**. When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of **\(** starting from the left. When the character **%** is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The **%** loses its special meaning when it is in a replacement string of more than one character or is preceded by a **\.**

A line may be split by substituting a new-line character into it. The new line in the *replacement* must be escaped by preceding it by **\.** Such substitution cannot be done as part of a **g** or **v** command list.

(.,.)t*a*

This command acts just like the **m** command, except that a *copy* of the addressed lines is placed after address *a* (which may be 0); "**.**" is left at the last line of the copy.

u

The **u**ndo command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent **a**, **c**, **d**, **g** , **i**, **j**, **m**, **r**, **s**, **t**, **v** , **G**, or **V** command.

(1,$)v/*RE/command list*

This command is the same as the global command **g** except the *command list* is executed with " **.**" initially set to every line that does *not* match the RE.

(1,$)V/*RE/*

This command is the same as the interactive global command **G** except the lines that are marked during the first step are those that do *not* match the RE.

(1,$)w*file*

The **w**rite command writes the addressed lines into the named file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless the *umask* setting [see *sh* (1)] dictates otherwise. The currently remembered filename is *not* changed unless *file* is the very first filename mentioned since *ed* was invoked. If no filename is given, the currently remembered filename, if any, is used (see **e** and **f** commands); the "**.**" is unchanged. If the command is successful, the number of characters written is typed. If *file* is replaced by **!**, the rest of the line is taken to be a shell [*sh*(1)] command whose standard input is the addressed lines. Such a shell command is *not* remembered as the current filename.

($)=

The line number of the addressed line is typed; the "**.**" is unchanged by this command.

!*shell command*

The remainder of the line after the **!** is sent to the *UNIX*® system shell [*sh*(1)] to be interpreted as a command. Within the text of that command, the unescaped character **%** is replaced with the remembered filename; if a **!** appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, **!!** repeats the last shell command. If any expansion is performed, the expanded line is echoed; "**.**" is unchanged.

(.+1)<new **line**>

An address alone on a line causes the addressed line to be printed. A new line alone is equivalent to "**.+1p** "; it is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, *ed* prints a **?** and returns to *its command level.*

Some size limitations: 512 characters per line, 256 characters per global command list, 64 characters per filename, and 128K characters in the buffer. The limit on the number of lines depends on the amount of user memory: each line takes one word.

When reading a file, *ed* discards ASCII NUL characters and all characters after the last new line. Files (for example, **a.out**) that contain characters not in the ASCII set (bit 8 on) cannot be edited by *ed.*

If the closing delimiter of an RE or of a replacement string (for example, **/**) is the last character before a new line, that delimiter may be omitted; in which case, the addressed line is printed. The following pairs of commands are equivalent:

s/s1/s2	**s/s1/s2/p**
g/s1	**g/s1/p**
?s1	**?s1?**

4. FILES

/tmp/e#	Temporary; # is the process number.
ed.hup	Work is saved in this file if the terminal is hung up.

5. DIAGNOSTICS

?	Command errors.
? *file*	Inaccessible file.
	(Use the **h**elp and **H**elp commands for detailed explanations).

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, *ed* warns the user if an attempt is made to destroy *ed 's* buffer via the **e orq** commands; it prints **?** and allows one to continue editing. A second **e** or **q** command takes effect at this point. The **-** command-line option inhibits this feature.

6. SEE ALSO

grep(1), sed(1), sh(1), stty(1)

7. CAVEATS AND BUGS

A **!** command cannot be subject to a **g** or a **v** command.

The **!** command and the **!** escape from the **e , r**, and **w** commands cannot be used if the editor is invoked from a restricted shell [see *sh*(1)].

The sequence **\n** in an RE does not match a new-line character.

The **l** command mishandles DEL.

Characters are masked to 7 bits on input.

EDOBJ

***** See Warning in Section 1.1 *****

1. NAME

edobj - Object file editor

2. SYNOPSIS

edobj [object[s]]

3. DESCRIPTION

Edobj is a Common Object File Format (COFF) file dumper and editor. *Edobj* performs three separate functions. It is an interactive dumper used for dumping file header, section header, optional header, relocation, line number, and symbol table information. It is an overwriter used for overwriting the actual hexidecimal values in the file with new values. It is a file editor used to change and update the values in the file header, section header, optional header, relocation, line number, and symbol table information.

One or more COFF files may be specified on the command line. The interactive dumper is entered by executing the *edobj* command. It provides the ability to dump out any portion of the COFF file in a readable format. The overwriter is accessible through the interactive dumper by executing the *O* command. The file editor is entered from the interactive dumper by executing the *E* command. The interactive dumper options are as follows:

! <command>
The *UNIX*® system command is to be executed.

?
The status of global variables is output.

a <symbol>
All of the occurrences of *symbol* are searched for in the symbol table and output.

b [number]
The number base is reset. If *number* is not specified, then the input number base is reset to the C standard. *Number* can be 8 for octal, 10 for decimal, and 16 for hexidecimal.

c

The current archive member is closed. The next archive member is opened.

d <sclass>

Sclass is a storage class. All of the symbols in the symbol table with the desired storage class are dumped out.

e <type>

All of the symbol table entries of a given *type* are output.

E

The object file editor is entered.

F <filename>

A search is performed to see if *filename* is a source file of the current object file.

f [ofilename]

The file header information is output. If an existing object file, *ofilename,* is specified, it becomes the currently opened file. Its file header information is then output.

g

The current object file is closed, and the next file specified on the command line is opened. If the current file is the only or last file specified on the command line, then the current file remains open.

h [section] [secndx] [*]

This command dumps out section header information. A "*" dumps all of the section header information in the current file. A particular section header is output by specifying a section name, *section* or a section index number, *secndx.* If no specifications are made, the last section header specified is output.

l [function] [symndx] [*]

Line number information is output. A "*" outputs all the line number entries. When a function is specified, the information for that function is output. If a symbol index is given, *symndx,* the line number information for the referenced symbol index is output. When the *l* command alone is given, the line number information for the last function specified will be output.

m [mcommand]

The command line menu is output. If a particular command is specified, only the menu information for that particular command is output.

n [count]

The symbol table information for the next *count* symbols is produced. If *count* is not specified it is assumed to be one.

o [a] [o] [p]

If no additional information is specified, then the entire optional header and patch list is dumped out. If *a* is specified, then the a.out header is output. *o* specifies the a.out with a library or pfile header. The patch list alone is specified by *p.*

O [address] [symbol ['line]]

The overwriter used to overwrite binary information in the object file is invoked. When specified, the overwriter starts at the given *address,* or at the address of the given *symbol.* If *symbol* is a function name, and *line* is an actual line number within that function, the overwriter can be specified to start at the address of a given line within a function.

r [section] [secndx] [*]

With the "*", all relocation information is dumped out. A specified *section,* dumps out the relocation information for a particular section. *Secndx,* a symbol index, outputs the relocation information for the referenced symbol index. If the *r* command alone is executed, the relocation information for the last specified section is output.

s [section [start][:end]]

Raw section data is dumped out. A *section* can be specified to output the raw data of a particular section. A range of section data can be specified. If a section is not specified, then the raw section data for the last specified section will be dumped out.

t [symbol [symbol]] [symndx [symndx]] [*]

The symbol table information is output. If no options are specified, then the information for the last specified symbol is output. A "*" outputs all symbol table entries. *Symbol* is used to specify a particular symbol. A range of information, from one symbol to another may be output by specifying two *symbols* or symbol indexes, *symndx .*

v [symbol [type]] [symndx [type]]

The value of a particular symbol is output. If *symbol* is specified, then the value of the named entry is output. A *type* may be specified. The *v* command alone outputs the value of the last symbol specified.

V

The version of **edobj** executing is printed along with the dmert generic object format it operates on.

q

The **edobj** session is terminated.

The overwriter provides the ability to make changes to the currently opened object file by overwriting the binary information in the file with new values. The overwrites can only be performed on codes found within sections. The overwriter commands are as follows:

\n

The next 16 bytes after the current line are displayed.

-

The previous 16 bytes before the current output line are displayed.

! <command>

The *UNIX*® system command is executed.

?

The status and content of certain global variables are output.

m

The program menu of the overwriter is listed.

o <[address] [symbol ['line]]>

Change the current address to *address,* and then output 16 bytes starting at the new address position. If a *symbol* is specified, the current address is changed to the start of the symbol, and the 16 bytes are displayed. If *symbol* is a function name and *line* is an actual line number within that function, the current address is changed to the location of the line within the function and the 16 bytes are displayed.

s <[hexnumber] ['char]>

A pattern is searched for starting at the current address. The pattern is made up of hexidecimal numbers or their character equivalent. If the pattern is found, the file pointer points to the start of the found pattern. The 16 bytes starting at this position are listed out. In each search, one byte of information must be specified. A hexidecimal value being searched for must start at the first half of a byte.

u

The last change made to the file is undone using this command.

hexnumber

A *hexnumber* can be any hexidecimal number. These hexidecimal numbers replace any of the displayed 16 bytes starting at the current address location. The desired changes are entered directly under the hexidecimal values to be changed. If a hexidecimal value is overwritten with a single space, ' ', the old value remains unchanged. At most, 16 bytes can be changed at any one time.

\tcharacters

The tab specifies the use of characters. The input characters replace the displayed characters and the hexidecimal representation of the characters within the object file. The replacement starts at the current address position. At most, 16 characters can be changed at any one time. If the user does not wish to overwrite a particular character, the character '.' is used to bypass the overwrite of that character.

q

The *overwriter* is exited, and control is turned back to the main portion of *edobj.*

The file editor portion of *edobj* provides the capability of making quick and easy changes to the COFF headers, relocation information, line number information, and symbol table entries. The portion of the COFF file to change is specified. The changes are then prompted. The time/date stamp in the file header is the only noneditable portion of the COFF file. The file editing commands are as follows:

! <command>

The *UNIX*® system *command* is to be executed.

f

The current file header is edited.

o [otype]

The optional file header is edited. The *otype* is *a* for an a.out header. It is *l* for a library header. It is *h* for a pfile header. The patch list is specified by *p .* A *u* specifies an update header. A *q* specifies a q file header.

h [[sectnum] [sectname]]

Editing is performed on a section header of the current file. The section number, *sectnum ,* or the name of the section, *sectname ,* must be specified to determine which particular section header to edit.

l [[symndx] [functname]]

Editing is performed on line number entries. The function the change is made in can be specified by its symbol table index, *symndx ,* or the function name, *functname .* The actual line entry to edit in the section that is prompted.

m

A menu of the available commands is output.

r [[sectnum] [sectname]]

A relocation entry in the current file is edited. The section which contains the relocation entry can be specified by *sectname ,* the name of the current section, or *sectnum ,* the number of the section. The individual relocation entry to be changed is prompted.

s [[sindex] [sname]]

An entry in the symbol table is edited. The symbol table index, *sindex ,* or symbol name, *sname ,* of the entry to change must be specified.

V

The version of **edobj** executing is printed along with the dmert generic object format it operates on.

q

The object file editing is terminated. Control is returned to the main portion of *edobj.*

4. CAVEATS

The tool cannot distinguish between a q file optional header and an update optional header. The user must know if he or she has a q file or an update optional header.

When searching for a section header or symbol name, if more than one entry exists with that name, the first section header or symbol table entry with that name is always retrieved.

A break executed in the file editing portion of *edobj* terminates *edobj.*

Editing changes which expand or contract the COFF file have side effects on other portions of the COFF file. Theses types of changes should be avoided.

EMACS

Software Release:
***** See Warning in Section 1.1 *****

1. NAME

EMACS - Interactive screen editor

2. SYNOPSIS

emacs [-l init_file] [+line_number] [file]

3. DESCRIPTION

Emacs invokes the EMACS interactive screen editor. The EMACS editor is used to construct and edit files on the *UNIX*® system. A window of text from the file being edited is displayed on the terminal screen, along with a status line describing the editor version and file being edited. Ordinary characters typed are inserted in the file, while escape sequences and control characters are used to invoke editing functions. Several files can be edited at the same time in different editing buffers, and two of the active buffers can be displayed on the same screen.

If given a *file* argument, EMACS reads the file into the buffer "Main". Otherwise, an empty buffer is created. If a *line_number* is given, EMACS moves to that line number in the specified file. If an *init_file* is specified, EMACS treats the contents of that file as EMACS commands to be executed on startup. EMACS also looks in your home directory for a file named *.emacs_init* and interprets commands from it before those in the specified *init_file* are executed. This option can also be specified with **.i**, in which case it suppresses the processing of *.emacs_init*. If an *.emacs_init* does not exist, EMACS looks in the EMACS library directory for a standard *.emacs_init*.

EMACS can be customized by the user through user-defined macro commands, which can redefine the effect of the basic editing commands. EMACS has a number of built in editing modes that customize some of the commands to support editing of particular types of files, such as C source programs or word processing source documents.

There are a number of self-help features in EMACS to aid in learning how to use the editor. Complete documentation appears in Tab 4 (EMACS Description).

4. MISCELLANEOUS CONVENTS

EMACS treats the following characters special in filenames:

$NAME	Substitute the environment variable NAME.
-USER	Substitute the home directory of USER.
-EMACS	Specifies the EMACS library directory. (Contains standard macros, etc.)
***,?**	Can be used for matching in partially specified filenames.
\`COMMAND\`	Substitute the output of running COMMAND.
^X	Control-X, where X is any character. These characters are input by holding down the control key and another key simultaneously on your terminal. EMACS also provides a special mode (controlify) to allow you to input control characters that your terminal cannot send to your system (see the discussion of modes).
{,[	Used as they are with the shell. EMACS uses the following notation to display and

input nonprinting characters:

Some of the control characters displayed are not very intuitive:

Some of the control characters displayed are not very intuitive:

^?	Rubout
^[	Excape
^]	Control-right-bracket
^\	Control-backslash
^_	Control underline
^@	Null
^H	Backspace (Displays as ^H when backspace mode is off; see succeeding paragraphs.)
^I	Tab (Displays as a ^I when notabs mode is on)
^J	Newline (Displays as ^J in searches)
^M	Carriage return
M-x	Meta-x, where x is any character (including a control character). Meta characters are entered by typing escape followed by another character.

Many EMACS commands take an optional numeric argument. The argument for a command precedes the command and can be specified a number of ways. Typing ^U specifies an argument of 4 or 4 times the current argument. Pressing the Escape key, followed by a sequence of digits with or without a leading '-', specifies a decimal value. The following are a few examples:

^U^U^N	Go forward 16 lines
M-123^N	Go forward 123 lines
M--12^N	Go forward -12 lines (goes back 12 lines).

Some EMACS commands prompt for a string parameter, such as a filename. Some of the normal EMACS commands can be used to edit the parameter while it is being entered. These include: **^F**,**^B**,**^D**,**^H**,**^A**,**^E**,**^K**,**^U**. In addition, typing the kill character (usually @) deletes the string, typing **^X** substitutes the contents of the current line in the buffer, and **^Y** substitutes the current filename. (The latter is a very convenient way of finding files with similar names.) Typing **^V** while typing a string causes EMACS to expand any shell meta characters (**$**,*****,**?**,etc.) in the string and show the first candidate in the result. If there is more than one expansion (for example, "foo.c" and "bar.c" for "*.c"), ^N and ^P enables the user to move backward or forward.

5. REGULAR EXPRESSIONS

EMACS uses an extension of the regular expression syntax used by **ed**(1) and **grep**(1) for regular expression searches and query replace. The following character sequences have special meaning:

.	Matches any single character except newline.
[xyz]	Matches one character among the set enclosed in brackets. (If the first character is **^**, it

	matches all but the specified characters.) If a **-** appears in the brackets, it designates a range of character values (that is, **[a-ez]** is equivalent to **[abcdez]**). The sequence **\n** can be used to specify a newline as one of the alternatives.
*****	Matches 0 or more of the preceding expression (a single character, specified as such or ***** or **[ ]**)
+	Matches 1 or more of the preceding expression
\{x,y\}	Matches **x** through **y** occurrences of the preceding expression. If **y** is omitted, it defaults to 255. If **x** is omitted it defaults to 0.
\(expr\)	Matches **expr** and saves the text so matches for later reference.
\(ex1\\|ex2\)	Matches expression **ex1** or **ex2** and saves the text matched for later reference. Note that any number of alternative expressions can be separated by **\ |**. This expression cannot be followed by *****, **+**, or a range to specify multiple matches.
\<	Matches 0 characters at the beginning of a word.
\>	Matches 0 characters at the end of a word.
^	Matches 0 characters at the beginning of a line
$	Matches 0 characters at the end of a line.
\n	Matches on newline at end of a line.

The following special sequences apply in the replacement strings to replace in "query replace".

&	Specifies the entire string matched by the from string.
%	Specifies the previous To string.
\n	Specifies the string matched by the n^{th} occurrence of **\(expr\)** (Regular expression replace only).
^J or \n	Specifies a newline is to be inserted at this point.

6. COMMAND SUMMARY

The following chart summarizes the available commands by category. Some commands appear in more than one category. Commands that are marked with ***** take a numeric argument that indicates how many times to do the command. Commands that are marked with **+** take a numeric argument that changes the behavior of the command in some other way.

6.1 General Commands

^G	Quit (Stops commands that prompt for things)
^Z	Exit one level (Usually exits EMACS)
^X^C	Exit EMACS
M-u	Undo. Undoes the result of the last modification
M-?	Help - Prompts for a command and displays its function.

M-w	Put a wall chart of command explanations in the current buffer
^L	Refresh the screen. (Argument indicates where to put the current line)

6.2 Character Oriented Commands

^F	Move forward one character
^B	Move backward one character
^D	Delete the character under the cursor
^H,^?	Delete the previous character
^T	Transpose the current and next character, move forward.
^C	Capitalize the current character

6.3 Word Oriented Commands

M-f	Move forward one word
M-b	Move backward one word
M-d	Delete forward one word
M-^?,M-^h	Delete backwards one word
M-c	Capitalize the next word
M-_	Underline the next word
M-a	Move to the beginning of the sentence
M-e	Move to the end of the sentence

6.4 Line Oriented Commands

^A	Move to beginning of line
^E	Move to end of line
^M-<	Move to beginning of file
^M->	Move to end of file
^P	Move back one line
^N	Move forward one line
M-g	Go to the line specified by the argument
^O	Create a blank line in front of the cursor
^J,^M	Make a new line (Just moves through empty lines).
^K	Kill (delete) to the end of line (with argument, kills whole lines)

6.5 Delete Commands

^D	Delete the character under the cursor
^H,^?	Delete the previous character
M-d	Delete forward one word
M-^?,M-^h	Delete backwards one word
^K	Kill (delete) to the end of line (with argument, kills whole lines)
^W	Delete the marked region (argument specifies a mark number)
^Y	Restore the last deletion (sets mark in front of it).
M-Y	Replace the marked region with the previous deletion (Use only immediately after **^Y**)

6.6 Display Commands

^L	Redraw the screen
^V	Display the next page
M-v	Display the previous page
M-<	Move to beginning of file
M->	Move to end of file
M-^L	Redraw with the current line at the top of the screen

6.7 Buffer Commands

Most prompt for a buffer name, entering return gets a list of active buffers.

^X^B	Change working buffer
^X^F	Find file. Changes buffer if file is in one; creates a new buffer and reads the file if not).
^X^K	Kill buffer
^X^N	Change buffer name (with argument, changes filename)
^X^T	Send region to buffer
^X=	Display statistics on buffer
^X2	Enter two window mode (prompts for buffer name for second window)
^X1	Make current window the only window
^X^O	Switch windows.

6.8 File Commands

^X^R	Read file into current buffer (with an argument, inserts the file at the current position)
^X^W	Write buffer to file (With an argument, appends to the file)
^X^S	Save current buffer into current file.
^X^F	Find file (does change buffer if file is in one, creates a new buffer and reads the file if not).
^X^N	Change buffer name (with argument, changes filename)
^X^L	Load macros from file. (With an argument, undefines all previously defined macros.)

6.9 Region Commands

M-	(Meta space) Set mark at position (argument is the mark number)
^X^X	Exchange mark and cursor position (argument is the mark number)
^W	Delete the region and put it on the kill stack
M-p	Put the marked region in the kill stack without deleting it.

6.10 Search and Replace Commands

(All prompt for search and replace strings.)

^S,^R	Forward and reverse incremental search. (Both display the string currently matched. **^S** moves to next occurrence, **^R** moves to previous occurrence. ^H deletes last character, **^G** quits search, escape exits search at currently displayed position. See regular expression search.
M-^S	Regular expression search. (Waits for whole expression to be typed). **^S** following **M-^S** goes to next occurrence.
M-r,M-^R	Ordinary and regular expression query replace. (Prompts at each occurrence of from string for the following:

y	Replace with "to" string and move on.
n	Do not replace this occurrence and move on.
r	Replace all of the rest, showing each replacement.
R	Replace the rest silently
p	Move to previous occurrence of from string.
.	Replace this one and stop.
<	Go back to the last replacement and stop.
^G	Quit query replace.
<escape>	Prompt for new "to" string, and replace this occurrence with it.

6.11 Window Commands

^X2	Enter two window mode (prompts for buffer name for second window).
^X1	Make current window the only window.
^X^O	Switch windows.
^X^^	Make current window grow by one line.

6.12 Special Input Commands

^Q	Takes the next input character and inserts it, even if it is a control character.
M-Q	Takes the next input character, makes it a meta character, and inserts it.
M-	Argument converted to an ascii character and inserts it.

6.13 Interaction with *UNIX*®

M-!	Prompts for a *UNIX*® command and executes it (with an argument, passes the buffer as standard input.
M-$	Execute *UNIX*® command, put output in buffer .exec
^X^D	Change working directory.
M-^M	Send the current buffer as mail. (Lines starting To: or Cc: are taken as destinations.)

6.14 Miscellaneous Commands

^X^M	Specifies modes (See the MODES paragraph)
M-s	Prints EMACS statistics
M-	Re-adjusts line boundaries in the whole buffer to fill lines evenly. (With an argument, it works only on the current region.
M-	Start a C comment.

7. MODES

Mode parameters enables the user to customize the behavior of certain commands. Some modes are switches, indicating only that something is either off or on; while others are numeric parameters. Modes can be set by the **^X^M** command. Typing **^X^M** followed by the name of a switch mode turns it on; typing **^U^X^M** followed by the name turns it off. To set numeric modes, give the value desired as an argument to **^X^M** (that is, M-500^X^Msave). Modes can be set automatically by putting **^X^M** commands in your .emacs_init. Modes can also be attached to a file by putting the string "EMACS_MODES: " followed by a list of mode settings in the first 10 lines of the file. (The mode settings can be preceded or followed by anything to allow the user to simulate a comment to other software processing the file.) The mode settings are separated by commas and can be of the following form:

modename	Set this switch mode.
!modename	Turn this switch mode off.

modename=x	Set this numeric parameter to x.

The following indicates the modes and their default settings. Switches are listed as either ON or OFF, while numeric parameters have specified values. Note that the system default .emacs_init may alter these settings on your local machine.

save	(OFF) Automatically saves each buffer after save-type characters of input or when you change buffers or run commands.
savetype	(256) Number of characters between automatic saves.
mailtype	(100) Number of characters between checks for new mail.
c	(OFF) Automatically indents during typing for C programs.
verbose	(ON) Provides prompts for meta and control-x commands.
fill	(ON) Automatically replaces a space with a newline when typing past the end of line or past fillcol characters.
fillcol	(72) Column beyond which lines are wrapped.
eofnl	(ON) Causes a newline to be appended to any file that does not end in one.
end_newline	(OFF) Causes attempts to move beyond the end of the file to add newlines.
keepscroll	(0) Number of lines kept from previous screen during **^V** and **M-v**.
smoothscroll	(OFF) Updates display via scrolling instead of jumping.
readonly	(OFF) Prevents saving the current buffer.
picture	(OFF) Enables 2-dimensional editing (See the manual)
tabstop	(8) Width of a tab character.
overwrite	(OFF) Causes input to replace characters already there, rather than insert.
nodelete	(OFF) Causes deletions to replace the characters with white space rather than deleting them.
rigid_newline	(OFF) Causes newline to always insert a new line, even if the next line is empty.
notabs	(OFF) Causes tabs to be expanded to spaces on input, and tabs in files to display as **^I**.
comcol	(40) Column where the **M-/** commands starts a comment.
backspace	(?) Causes backspaces to appear as cursor motion, not **^H**. This mode is set ON if your terminal handles underscored characters, OFF otherwise.
nobell	(OFF) Causes EMACS not to ring the terminal bell on an error.
caseless	(OFF) Causes all searches to ignore upper/lower case distinctions.
usilent	(OFF) Causes output from *UNIX*® commands run from EMACS to be discarded.
noecho	(OFF) Causes output from **M-$** commands not to be echoed.

controlify	(OFF) Causes a sequence of ctl_char followed by another character to translate into the second character made a control character.
ctl_char	(30) Prefix for controlify (This is an integer specifying the ascii code of the character, the default is ^^.
lnumb	(ON) Displays line numbers.
lnowid	(4) Width of line numbers.
time	(OFF) Displays a clock.
display_percent	(OFF) Displays current position as a percentage of the whole file.
flow_lim	(0) If non-zero, flow control is enabled whenever flow_lim characters are sent to the terminal in a burst.
height	(?) Height of screen area for buffer display (set automatically).
width	(?) Width of screen.
tspeed	(?) Describes the terminal to host speed.
autoload	(ON) Automatically loads macros when first referenced.
leftmargin	(0) In picture mode, this is the number of characters to the left of the screen.
display_euc	(?) Determines whether characters with the high order bit are considered extended ASCII or EMACS meta-characters. (Set on if your terminal displays them)

8. ENVIRONMENT

The variable *MAIL* is the filename that EMACS looks at for newly arrived mail. If your mail is forwarded to some other system, MAIL should not be exported. The environment variable MAILER optionally specifies the name of a mail command to use for sending mail. The environment variable SHELL specifies the shell to use to execute shell commands. If the environment variable TEMP4EMACS is set, it is taken as the pathname of a directory in which the temporary files of EMACS should placed.

9. FILES

$HOME/.emacs_init

EMACS/.emacs_init

$HOME/emacs[0-11]

EMACS/macros

EMACS/terminals

EMACS/helpfile

EMACS/errfile

The .emacs_init files, if present, contain a standard set of initializations to be made when EMACS is started . The characters in the file are interpreted as if entered as commands from the terminal. The most common application of this is to set modes different from the default modes.

When EMACS is killed by an internal error, the **kill**(1) command, or by hanging up during an editing session, it saves your buffers in the files emacs0-emacs11 in the home directory. **Mail**(1) is received as notification of what buffers were saved.

The directory EMACS is the EMACS library, the location of which depends on the local installation. Pathnames starting with EMACS are translated to the local path of this directory by EMACS but *not* by other tools. EMACS/macros may contain macro packages that can be loaded. EMACS/terminals contains terminal information for EMACS. The other two files contain internal data for EMACS.

ENV

***** See Warning in Section 1.1 *****

1. NAME

ENV - Set environment for command execution

2. SYNOPSIS

env [-] [name=value] ... [command args]

3. DESCRIPTION

Env obtains the current *environment*, modifies it according to its arguments, then executes the command with the modified environment. Arguments of the form *name* = *value* are merged into the inherited environment before the command is executed. The **-** flag causes the inherited environment to be ignored completely so that the command is executed with exactly the environment specified by the arguments.

If no command is specified, the resulting environment is printed, one name-value pair per line.

4. SEE ALSO

sh(1)

ERRPORT

***** See Warning in Section 1.1 *****

1. NAME

errport - Interface to craft output spooler

2. SYNOPSIS

/bin/errport

3. DESCRIPTION

The error logging process, *errport*, is a 2-process system that receives error messages from both supervisor and kernel processes and the kernel. *Errport* adds a control string if necessary and forwards the error messages to the output spooler process for output on one or more of the craft terminals or log files.

Interfaces for the kernel, kernel processes, and supervisor processes to the *errport* system are provided in library *errpt* (see *Volume 4, System Libraries*, Chapter 7, "Operating System Libraries").

The kernel formats and sends a message to the system port PT_ERRLOG. The FIFO driver (a kernel process */prc/fda*) attaches to this port and receives these messages. Their text is stored in an area of low core (starting at physical address Ox9000). A *UNIX*® process (*/bin/errport*) reads this area by trapping to the FIFO driver. The text of each message it finds is placed in a message destined for the craft spooler.

Messages sent to PT_ERRLOG and received prior to a bootstrap of any kind are saved across the bootstrap.

Both processes are created at boot time. The FIFO driver is *pcreated* at boot time and ULARP brings up the *UNIX*® process. This process then opens the low core FIFO (*/dev/errport*), which establishes the communication between the two processes.

EXPR

***** See Warning in Section 1.1 *****

1. NAME

expr - Evaluate arguments as an expression

2. SYNOPSIS

expr arguments

3. DESCRIPTION

DESCRIPTION

Expr evaluates the arguments as an expression. After evaluation, the result is written on the standard output. Terms of the expression must be separated by blanks. Characters special to the shell must be escaped. Note that **0** is returned to indicate a zero value, rather than the null string. Strings containing blanks or other special characters should be quoted. Integer-valued arguments may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, 2s complement numbers.

The following list contains operators and keywords. Characters that need to be escaped are preceded by \. The list is in order of increasing precedence with equal precedence operators grouped within **{}** symbols.

expr **\\ |** *expr*
: Returns the first *expr* if it is neither null nor **0**; otherwise, returns the second *expr*.

expr **\\&** *expr*
: Returns the first *expr* if neither *expr* is null nor **0**; otherwise returns **0**.

expr { =, \\>, \\>=, \\<=, != } *expr*
: Returns the result of an integer comparison if both arguments are integers; otherwise, returns the result of a lexical comparison.

expr { **+**, **-** } *expr*
: Addition or subtraction of integer-valued arguments.

expr { \\ *, /, % } *expr*
: Multiplication, division, or remainder of the integer-valued arguments.

expr : *expr*
: The matching operator **:** compares the first argument with the second argument which must be a regular expression; regular expression syntax is the same as that of *ed* (1) except all patterns are "anchored" (that is, begin with **^**) and, therefore, **^** is not a special character in that context. Normally, the matching operator returns the number of characters matched (**0** on failure). Alternatively, the **\\(...\\)** pattern symbols can be used to return a portion of the first argument.

4. EXAMPLES

(1) *a*=**'expr $***a* **+ 1'**

Adds 1 to the shell variable **a**.

(2) **# '** for **$a** equal to either "/usr/abc/file" or just "file",
expr $a : '.*/ \ (.*\ )' \| $a

Returns the last segment of a pathname (that is, **file**). Watch out for **/** alone as an argument; *expr* takes it as the division operator (see BUGS).

(3) # A better representation of example 2.
expr //$a : '.*/ \ (.* \ )'

The addition of the **//** characters eliminates any ambiguity about the division operator and simplifies the whole expression.

(4) expr $VAR : '.*'

Returns the number of characters in *$VAR .*

5. SEE ALSO

ed(1), sh(1)

6. EXIT CODE

As a side effect of expression evaluation, *expr* returns the following exit values:

0	If the expression is neither null nor **0**.
1	If the expression *is* null or **0**.
2	For invalid expressions.

7. DIAGNOSTICS

syntax error	For operator/operand errors.
non-numeric argument	If arithmetic is attempted on such a string.

8. BUGS

After argument processing by the shell, *expr* cannot tell the difference between an operator and an operand except by the value. If **$a** is an **=** , the command **expr $ a = '='** looks like **expr = = =** as the arguments are passed to *expr* and are taken as the **=** operator. The following command works:

```
expr X$a = X=
```

FALLOC

***** See Warning in Section 1.1 *****

1. NAME

falloc - Allocate a contiguous file

2. SYNOPSIS

falloc Filename size

3. DESCRIPTION

Falloc allocates a contiguous file with the specified filename and size (512-byte blocks).

4. DIAGNOSTICS

The command complains that a needed directory is not searchable, the final directory is not writable, the file already exists, or there is not enough space for the file.

FALSE

*** See Warning in Section 1.1 ***

1. NAME

false

3. DESCRIPTION

See *true*.

FGREP

***** See Warning in Section 1.1 *****

1. NAME

fgrep - See *grep*.

3. DESCRIPTION

See *grep*.

FIND

***** See Warning in Section 1.1 *****

1. NAME

find - find files

2. SYNOPSIS

find pathname-list expression

3. DESCRIPTION

Find recursively descends the directory hierarchy for each pathname in the *path-name-list* (that is, one or more pathnames) seeking files that match a boolean *expression* written in the following primaries. In the descriptions, the argument *n* is used as a decimal integer where +*n* means more than *n*, -*n* means less than *n* , and *n* means exactly *n*.

-name *file*	True if *file* matches the current filename. Normal shell argument syntax may be used if escaped (watch out for [, ? and *).
-perm *onum*	True if the file permission flags exactly match the octal number *onum* [see *chmod* (1)]. If *onum* is prefixed by a minus sign, the flags are compared: (flags&onum)==onum.
-type *c*	True if the type of the file is *c* , where *c* is **b, c, d, p**, or **f** for block special file, character special file, directory, fifo (a.k.a named pipe), or plain file.
-links *n*	True if the file has *n* links.
-user *uname*	True if the file belongs to the user *uname*. If *uname* is numeric and does not appear as a login name in the **/etc/passwd** file, it is taken as a user ID.
-group *gname*	True if the file belongs to the group *gname* . If *gname* is numeric and does not appear in the **/etc/group** file, it is taken as a group ID.
-size *n*	True if the file is *n* blocks long (512 bytes per block).
-atime *n*	True if the file has been accessed in *n* days.
-mtime *n*	True if the file has been modified in *n* days.
-ctime *n*	True if the file has been changed in *n* days.
-exec *cmd*	True if the executed *cmd* returns a zero value as exit status. The end of *cmd* must be punctuated by an escaped semicolon. A command argument **{ }** is replaced by the current pathname.
-ok *cmd*	Like **-exec** except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing **y**.
-print	Always true; causes the current pathname to be printed.
-cpio *device*	Write the current file on *device* in *cpio* (1) format (5120-byte records).
-newer *file*	True if the current file has been modified more recently than the argument *file* .
(*expression*)	True if the parenthesized expression is true (parentheses are special to the shell and must be escaped).

The primaries may be combined using the following operators (in order of decreasing precedence):

1) The negation of a primary (**!** is the unary *not* operator).

2) Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries).

3) Alternation of primaries (**-o** is the *or* operator).

4. EXAMPLE

To remove all files named **a.out** or ***.o** that have not been accessed for a week, type the following example:

```
find / \( -name a.out -o -name '*.o'\) -atime +7 -exec rm { }\;
```

5. FILES

/etc/passwd,/etc/group

6. SEE ALSO

chmod(1), cpio(1), sh(1)

FMOVE

***** See Warning in Section 1.1 *****

1. NAME

fmove - Make a file multiextent

2. SYNOPSIS

fmove filename

3. DESCRIPTION

Fmove makes the specified file into a multiextent file.

4. DIAGNOSTICS

The command complains a needed directory is not searchable; the final directory is not writable; the file does not already exist; the file is open; or there is not enough contiguous space for the file.

FSAUDIT

***** See Warning in Section 1.1 *****

1. NAME

fsaudit - File system mount auditor

2. SYNOPSIS

fsaudit <SG database name><partition name>[mount point]

3. DESCRIPTION

DESCRIPTION

Fsaudit locates the given *partition name* in the given *SG database,* and extracts the disk number, partition number, partition block count, and partition inode block count for that partition. This information is then used to initialize the named partition, mounting it on the optional *mount* point. If no mount point name is given, then */dev/<partition name>* is used. *Fsaudit* basically does an incremental mount of the named file system. If the file system is mounted, *fsaudit* returns immediately. Otherwise, the file system is mounted on the named device file. If the device file does not exist, then it is created. *Fsaudit* is similar to *fsinit,* but this command does not destroy the contents of the file system.

4. DIAGNOSTICS

Fsaudit fails with an error code if the named SG database cannot be attached or read, or if the requested partition cannot be successfully mounted. Appropriate error messages are printed before *fsaudit* exits.

5. FILES

/etc/fsaudit

6. EXAMPLES

The following command reads the application SG database (appdmert.sg), extracts the information needed to mount no5text, and mount the no5text partition on /tmp/no5mtpt:

```
fsaudit /database/appdmert.sg no5text /tmp/no5mtpt
```

The following command acts the same as the preceding command, but the no5text partition is mounted on /dev/no5text by default:

```
fsaudit /database/appdmert.sg no5text
```

7. CAVEATS

Note that *fsaudit* can not always mount a corrupted file system; in which case, *fsinit* may have to be used with discretion.

8. SEE ALSO

fsinit(1)

FSDB

***** See Warning in Section 1.1 *****

1. NAME

fsdb - File system debugger

2. SYNOPSIS

/etc/fsdb special [-]

/etc/fsdb [-w] special [-]

3. DESCRIPTION

Fsdb can be used to patch up a damaged file system. It contains conversions which translate block and i-numbers into their corresponding disk addresses. It also includes mnemonic offsets which can be used to access different parts of an i-node.

These features simplify correcting control block entries or descending the file system tree.

Fsdb contains several error checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking *fsdb* with the optional - argument or by the use of the 0 symbol. (*fsdb* reads the i-size entries from the super-block of the file system in order to perform these checks.)

The **-w** option enables *fsdb* to write to mounted file systems. Without **-w**, *fsdb* defaults to the read only mode. To patch a mounted file system, the *openwd* command must have been issued prior to *fsdb* (see *openwd* in this section). *Openwd* allows *fsdb* to open special device files for mounted file systems with read/write.

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. Hexadecimal numbers must be prefixed by either x or 0x and must be terminated by a colon. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

Since *fsdb* reads a block at a time, it is able to handle both raw and block I/O. A buffer management routine retains commonly used blocks of data and reduces the number of read system calls. Some assignment operations result in a delayed write-through of the corresponding block.

These symbols are recognized by *fsdb*:

i	Convert from i-number to i-node address
b	Convert to block address
d	Directory slot offset
+,-,*,/	Address arithmetic
q	Quit
>,<	Save, restore an address
=	Numerical assignment
+=	incremental assignment
-=	Decremental assignment

="	Character string assignment
O	Error checking flip-flop
p	General print facilities
f	File print facility
B	Byte mode
W	Word mode
S	Half-word mode
!	Escape to shell.

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the delete character. If a number follows the *p* symbol, that many entries are printed. A check is made to detect block boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed. The following print options are available:

i	Print as i-nodes
d	Print as directories
o	Print as octal half words
e	Print as decimal words
c	Print as characters
b	Print as hexadecimal bytes
h	Print as hexadecimal words.

The *f* symbol is used to print data blocks associated with the current i-node. (Blocks are numbered starting with zero.) The desired print option letter follows the block number or the *f* symbol. It checks for special devices and for nonzero block pointers.

Dots, tabs, and spaces may be used as function delimiters but are not necessary. A line which contains only a newline character increment S the current address by the size of the data type last printed. That is, the address is set to the next byte, word, half-word, directory entry or i-node, allowing the user to step through a region of a file system. Information is printed in a format appropriate to the data type. Bytes, words, and double words are displayed with the hexadecimal address followed by the value in hexadecimal and decimal. A ".B" or ".S" is appended to the address for byte and half-word values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. I-nodes are printed with the labeled fields describing each element. The following mnemonics are used for i-node examination and refer to the current working i-node:

md	Mode
ln	Link count
&	User id number
gid	Group id number
sz	File size
a#	Data block numbers (0-7)
at	Access time
mt	Modification time

maj	Device class number
min	Logical device identification number.

4. EXAMPLES

Examples of *fsdb* uses are as follows:

386i	Prints i-number 386 in an i-node format. This now becomes the current working i-node.
a0b.p0x10:h	Prints the first 16 words of the file for which a 0 is the starting block number.
2i.fd	Prints the first 32 directory entries for the root i-node of this file system.
d5i.fc	Changes the current i-node to the i-node associated with the fifth directory. The first 512 bytes of the file are then printed in ASCII.
lb.p0o	Prints the superblock of this file system in octal.

5. LIMITATIONS

The *fsdb* is a useful command which would be even more useful if it worked as planned (for example, the = operator when applied to an i-node value).

FSINIT

***** See Warning in Section 1.1 *****

1. NAME

fsinit - File system initializer

2. SYNOPSIS

fsinit <SG database name><partition name>[mount point]

3. DESCRIPTION

Fsinit locates the given *partition name* in the given *SG database,* and extracts the disk number, partition number, partition block count, and partition inode block count for that partition. This information is then used to initialize the named partition, mounting it on the optional *mount point.* If no mount point name is given, then */dev/<partition name>* is used. *Fsinit* uses the *clrfs* system call to initialize the super block of the named partition with the data extracted from the named SG database. Basically, *fsinit* acts like a smart *"mount"* command; it tries to guarantee that the named partition is successfully mounted and is correctly initialized as per the SG database specifications.

4. DIAGNOSTICS

The *Fsinit* fails with an error code if the named SG database cannot be attached or read, or if the requested partition cannot be successfully mounted. Appropriate error messages are printed before *fsinit* exits.

5. FILES

/etc/fsinit

6. EXAMPLES

The following command reads the application SG database (appdmert.sg), extracts the information needed to initialize the super block of no5text, clears and updates the no5text super block, using the new information, and mounts the no5text partition on /tmp/no5mtpt:

```
fsinit /database/appdmert.sg no5text /tmp/no5mtpt
```

The following command acts the same as the preceding command, but the no5text partition is mounted on /dev/no5text by default:

```
fsinit /database/appdmert.sg no5text
```

.

7. CAVEATS

Note that *fsinit* wipes out the existing information in the super block of the named partition; thereby, making that partition appear "empty"□as any link to the information stored therein is lost. For this reason, *fsinit* is used with discretion and care.

8. SEE ALSO

fsaudit(1)

FSIZE

***** See Warning in Section 1.1 *****

1. NAME

fsize - Get size of contiguous file

2. SYNOPSIS

fsize filename

3. DESCRIPTION

Fsize returns the size of specified file.

4. LIMITATIONS

Fsize is meaningless on noncontiguous files.

GREP

***** See Warning in Section 1.1 *****

1. NAME

grep, fgrep - Search a file for a pattern

2. SYNOPSIS

grep [options] expression [files]

fgrep [options] strings [files]

3. DESCRIPTION

Commands of the *grep* family search the input *files* (standard input default) for lines matching a pattern. Normally, each line found is copied to the standard output. *Fgrep* patterns are fixed *strings* and are fast and compact. *Grep* patterns are limited regular *expressions* in the style of *ed* (1) and uses a compact nondeterministic algorithm. The following *options* are recognized:

-v	All lines but those matching are printed.
-x	(Exact) only lines matched in their entirety are printed (*fgrep* only).
-c	Only a count of matching lines is printed.
-l	Only the names of files with matching lines are listed (once), separated by new lines.
-n	Each line is preceded by its relative line number in the file.
-b	Each line is preceded by the block number on which it was found. This is sometimes useful in locating disk block numbers by context.
-s	The error messages produced for nonexistent or unreadable files are suppressed (*grep* only).
-e *expression*	Same as a simple *expression* argument, but useful when the *expression* begins with a **-** (does not work with *grep*).
-f *file*	The *strings* list (*fgrep*) is taken from the *file* .

In all cases, the filename is output if there is more than one input file. Care should be taken when using the characters **$, *, [, ^, |, (,)**, and \ in *expression* because they are also meaningful to the shell. It is safest to enclose the entire *expression* argument in single quotes (**'...'**).

Fgrep searches for lines that contain one of the *strings* separated by new lines.

4. SEE ALSO

ed(1), sed(1), sh(1)

5. DIAGNOSTICS

Exit status is 0 if any matches are found, 1 if none, and 2 for syntax errors or inaccessible files (even if matches were found).

6. BUGS

Ideally there should be only one *grep*, but a single algorithm that spans a wide enough range of space-time tradeoffs does not exist.

Lines are limited to 256 characters; longer lines are truncated.

ICHK

***** See Warning in Section 1.1 *****

1. NAME

ichk - File system consistency check and interactive repair

2. SYNOPSIS

ichk [-bnsvy] [filesystem] ...

3. DESCRIPTION

Ichk audits DMERT file systems for consistency and allows the operator to correct any discrepancies. Note that a file system must be unmounted before corrections can be made (see *LIMITATIONS*). Since these corrections, in general, result in a loss of data, the program requests operator concurrence for each such action. All questions should be answered by typing "yes" or "no" followed by a newline character. Typing "yes" causes the correction to take place. However, if the program does not have write permission on the file system, all questions are automatically answered "no".

The following options are recognized:

-b	Runs only phases that check for block inconsistencies (phases 1, 2, and 7).
-n	Causes all questions to be answered "no" automatically.
-s	Salvages all blocks not found in some file and makes a new free list (if the bit map is bad, it too is recreated).
-v	Turns the verbose mode off.
-y	Causes all questions to be answered "yes" automatically.

The program consists of seven separate phases. Some phases are skipped if they are not needed. In phase one, *ichk* examines all block pointers in all files; checking for pointers which are outside of the file system (BAD), for blocks which appear in more than one file (DUP), and for blocks which are (incorrectly) included in the i-node's block pointer list but are actually beyond the size of the file. A table is made of all DUP blocks and all defective files are marked for clearing (BAD and DUP only). No correction takes place in this phase for BAD and DUP errors. However, the operator is given an option of clearing the incorrect (BEYOND FILE SIZE) block pointers by replacing them with null (0) values.

The second phase is run only if DUP blocks were found in phase one. This phase finds the rest of the DUP blocks and marks them for clearing.

If the **-b** option has not been selected, the third and fourth phases check the directory structure of the file system by descending the directory tree and examining each entry. A count of the number of references to each file is maintained. If any entry refers to an unallocated file, a file marked for clearing, or a file number outside the file system, the entry is printed; if the operator agrees, it is removed. Refusing to remove an entry to a marked file clears the mark and preserves the file with its subsequent entries.

If the **-b** option has not been selected, then phase five lists all marked or unreferenced files. With concurrence from the operator, each of these files is then cleared. In addition, any file with a link count that does not agree with the number of references listed and the operator agrees, the link count is adjusted.

If the salvage option, **-s**, and/or the block option, **-b**, is on, phase six is skipped. Otherwise, the bit map and free list are examined. If any blocks are found outside of the file system and have been previously encountered in a file or elsewhere in the free list or do not appear in the bit map, free list, or any file, then the list or bit map is pronounced BAD and a salvage is called. Operator agreement sets the salvage option and proceeds to the next phase. If there are no defects in the free list and bit map and all blocks are accounted for, the check is finished. Otherwise, the number of missing blocks is printed, and a salvage is requested.

The last phase is the salvage operation, where the free list and/or the bit map is recreated. It is run whenever the salvage option is on or a problem has been found in phase 6. Simply stated, a new free list is constructed containing all blocks not found in some file (if the bit map is bad, it too is recreated).

The system responses are, in general, self-explanatory and follow the sequence described previously. In the description of the output which follows, this notation is used:

<b>	Block number
<i>	I-node number
<fname>	File pathname
<n>	Positive integer
<c>	Option character

Ichk begins with the following output.

```
<filesystem> { ( NO WRITE ) }
```

3.1 Phase 1 Check Blocks

The (NO WRITE) message indicates that the program does not have write permission on the file system. Therefore, subsequent corrections are suppressed by automatically answering "no" to all questions. Phase 1 then proceeds to list any BAD, DUP, or BEYOND FILE SIZE blocks and their i-node number, as follows:

```
<b>                 BAD                         I=<i>
<b>                 DUP                         I=<i>
<b>                 EXCESSIVE DUPS              I=<i>
<b>                 BEYOND FILE SIZE            I=<i>
```

If a -[**yn**] option is not specified, the program waits for a response of "yes" or "no" after each BEYOND FILE SIZE message. A "yes" answer replaces the block pointer with a null (0) value. A "no" answer leaves the block pointer unaffected. In either case, all other data associated with the i-node is not affected, nor is the i-node cleared.

If too many DUPs are encountered, the program lists all blocks, but it does not mark the excess DUPs for later processing. When phase 1 is finished, if any DUPs were encountered, phase 2 is run. Otherwise, phase 2 is skipped.

3.2 Phase 2 Rescan for More

```
<b> DUP              I = <i>
```

Check descends the directory tree asking whether to remove any defective entries.

3.3 Phase 3 Check Pathnames

No text.

3.4 Phase 4 Check Connectivity

```
I OUT OF RANGE             I = <i>            <fname>            REMOVE?
```

UNALLOCATED	I = **<i>**	<fname>	REMOVE?
BAD/DUP	I = **<i>**	<fname>	REMOVE?

If no option has been specified, the program waits for a response of "yes" or "no" after each question. A "no" answer to the BAD/DUP entry unmarks that i-node for clearing. This suppresses any subsequent correction to that file.

3.5 Phase 5 — Check Reference Counts

Now *ichk* clears or adjusts any defective files. Again, if no option has been specified, it waits for a "yes" or "no" response to each question. The program also indicates whether each entry is a file or a directory.

UNREFERENCED	{FILE/DIRECTORY}	I = **<i>**	CLEAR?
BAD/DUP	{FILE/DIRECTORY}	I = **<i>**	CLEAR?
LINK COUNT	{FILE/DIRECTORY}	I = **<i>**	ADJUST?

3.6 Phase 6 — Check Free List

If the salvage option is not on, the program now validate the free list and bit map. Otherwise, this phase is skipped. If there are any errors, it specifies them and requests a salvage.

BAD FREE CHAIN	SALVAGE?
BAD BIT MAP	SALVAGE?
FREE LIST BOTCH	SALVAGE?

3.7 Phase 7 — Salvage Free List

This operation is performed only upon request.

The following totals are printed:

(1) Number of allocated files including directories and special files

(2) Number of blocks in use

(3) Number of blocks in the free list.

<n> FILES	**<n>** BLOCKS	**<n>** FREE

If the file system has been modified, then the message, BOOT UNIX(NO SYNC!), is printed and the program goes into a loop. This is only a reminder to the operator since the program can be forced to terminate with a <DEL> character.

4. DIAGNOSTICS

Exit code 0 is returned for no file system consistency errors; 1, if file system errors were found and all were corrected; 2, if file system errors were found but not all were corrected; and 3, for *ichk* process failure before completion (for example, because of inability to open device). A number of errors can terminate *ichk*. An illegal option is ignored, but the inability to open the file system is shown as follows:

CAN NOT OPEN **<filesystem>**

An I/O error on the file system also returns an error message. In this case, the operator is given the choice of exiting (yes) or continuing (no). This error is generally a hardware error and continuing is rarely a good idea.

CAN NOT READ **<filesystem>**	BLOCK **<b>** EXIT?
CAN NOT SEEK **<filesystem>**	BLOCK **<b>** EXIT?
CAN NOT WRITE **<filesystem>**	BLOCK **<b>** EXIT?

5. LIMITATIONS

The *ichk* software has been known to produce core images on large file systems.

Since *ichk* diagnoses problems that exist on the disk version of the file system (such as superblock and bit map), two warnings are given.

-File system errors are found by

The *ichk* valid only if the file system was unmounted during the *ichk* process. If the file system cannot be unmounted and an *ichk* is mandatory then the operator should run several (three or more) *ichk* with *-n* option on a quiet (if possible) mounted file system. Any errors that appear in all the *ichks* probably exist. If the errors disappear in all the *ichks,* they are probably temporary and of no concern.

-Corrections that

ichk makes to the mounted file system are overwritten (invalidated) with in-core file system information. Therefore, corrections should be made only when the file system is unmounted.

ID

***** See Warning in Section 1.1 *****

1. NAME

id - Print user and group IDs and names

2. SYNOPSIS

id

3. DESCRIPTION

Id writes a message on the standard output giving the user and group IDs and the corresponding names of the invoking process. If the effective and real IDs do not match, both are printed.

KILL

***** See Warning in Section 1.1 *****

1. NAME

kill - Terminate a process

2. SYNOPSIS

kill [- signo] PID ...

3. DESCRIPTION

Kill sends signal 15 (terminate) to the specified processes. This command normally kills processes that do not catch or ignore the signal. The process number of each asynchronous process started with **&** is reported by the Shell (unless more than one process is started in a pipeline, in which case the number of the last process in the pipeline is reported). Process numbers can also be found by using *ps* (1).

If process number 0 is specified, all processes in the process group are signaled.

The killed process must belong to the current user unless the user is the super user.

If a signal number preceded by **-** is given as first argument, that signal is sent instead of terminated. In particular "kill -9 ..." is a sure kill.

SIGNALS

1	Hang-up
2	Interrupt
3	Quit
4	Illegal instruction
5	Trace trap
6	IOT instruction
7	EMT instruction
8	Floating point exception
9	Kill (cannot be caught or ignored)
10	Bus error
11	Segmentation violation
12	Bad argument to system call
13	Write on a pipe with no one to read it
14	Alarm clock
15	Software termination signal

4. SEE ALSO

ps(1), sh(1)

KILLP

***** See Warning in Section 1.1 *****

1. NAME

killp - Terminate user processes by invoking pathname

2. SYNOPSIS

killp [-signo] full-pathname

3. DESCRIPTION

Killp sends **signo** to all processes with an invocation path of **full-pathname**. **signo** is a decimal integer constant.

If **signo** is defaulted, the signal 9 (**SIGKILL**) is sent.

The path **full-pathname** must begin with a slash (/).

4. SEE ALSO

kill(1), pkill(1), ps(1)

5. DIAGNOSTICS

If an error occurs, a message is written to *stderr* and an exit status of 1 is returned.

6. LIMITATIONS

Killp only kills processes whose argv[0] is identical to the full pathname with which the process was invoked. Therefore, processes invoked with relative pathnames or processes that modify argv[0] are not killable with this command. Also, processes invoked from linked files are only killed if argv[0] is the full pathname of the appropriate linked filename (that is, the **full-pathname** argument to *killp*).

7. CAVEATS

The *killp* command does not terminate a suspended process because it results in a message requesting the process, to kill itself. Since the process is suspended, it cannot handle these messages.

LF

***** See Warning in Section 1.1 *****

1. NAME

lf - List contents of directory

2. SYNOPSIS

lf [-1aAbcCdfFgilmnopqrRstuvVx] [files ...]

3. DESCRIPTION

For each directory argument, *lf* lists the contents of the directory; for each file argument, *lf* repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are several major listing formats. The format chosen depends on whether the output is going to a teletype and may also be controlled by option flags. The default format for a teletype is to list the contents of directories in multi-column format, with the entries sorted down the columns. (Files which are not the contents of a directory being interpreted are always sorted across the page rather than down the page in columns. This is because the individual filenames may be arbitrarily long. This format may be requested with -x.) If the standard output is not a teletype, the default format is to list one entry per line. (This is the -1 option.) A long listing, detailing much information about the file, can be requested with the -l option. Finally, there is a stream output format in which files are listed across the page, separated by `,' characters. The -m flag invocation enables this format.

There are an unbelievable number of options:

-1	Force one entry per line output format, for example, to a teletype.
-a	List all entries; usually entries beginning with '.' are suppressed.
-A	List all entries except '.' and '..'
-b	Force printing of non-graphic characters to be in the \ddd notation, in octal.
-c	Use time of file creation for sorting or printing.
-C	force multi-column output, for example, to a file or a pipe.
-d	If argument is a directory, list only its name not its contents (mostly used with -l to get status on directory.)
-f	Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off -l, -t, -s, and -r, and turns on -a; the order is the order in which entries appear in the directory.
-F	Cause directories to be marked with a leading and trailing square brackets '[dir]' and executable files to be marked with a leading '*'; this is the default.
-g	Same as -l except the owner name is not printed.
-i	Print i-number in first column of the report for each file listed.
-l	List in long format, giving mode, number of links, owner, group, size in bytes, and time of last modification for each file. If the file is a special file, the size field contains the major and

	minor device numbers instead.
-m	Force stream output format.
-n	Same as -l except owner and group are printed as integers instead of searching the password and group files for translations.
-o	Same as -l except the group name is not printed.
-p	Same as -F except executable files are not marked.
-q	Force printing of non-graphic characters in filenames as the character \`?'; this normally happens only if the output device is a teletype.
-r	Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.
-R	Recursively list subdirectories encountered.
-s	Give size in blocks, including indirect blocks, for each entry.
-t	Sort by time modified (latest first) instead of by name, as is normal.
-u	Use time of last access instead of last modification for sorting (-t) or printing (-l).
-v	On multi-column output, column width is fixed (non-variable). This is the default.
-V	Print version number and exit.
-x	Force columnar printing to be sorted across rather than down the page; this is the default.

The mode printed under the -l option contains 11 characters which are interpreted as follows:

b	If the entry is a block-type special file
c	If the entry is a character-type special file
d	If the entry is a directory
p	If the entry is a FIFO (named pipe) special file
-	If the entry is a plain file
C	If the entry is a one contiguous extents file
x	If the entry is allocated by contiguous extents.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next to permissions to others in the same user-group; and the last to all others. Within each set the three characters indicate permission respectively to read, write, or execute the file as a program. For a directory, \`execute' permission is interpreted to mean permission to search the directory for a specified file. The permissions are indicated as follows:

r	If the file is readable
w	If the file is writable
x	If the file is executable

-	if the indicated permission is not granted.

The group-execute permission character is given as **S** if the file has set-group-ID mode; likewise, the user-execute permission character is given as **S** if the file has set-user-ID mode. An S in either position indicates the set-ID bit is set but the corresponding execute bit is not.

The last character of the mode (normally \`x' or \`-') is **T** if the 1000 (save text) bit of the mode is ON. A **T** in this position indicates the save text bit is on but there is no execute permission for 'other.' See chmod(1) for the meaning of this mode.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

4. FILES

/etc/passwd
/etc/group

5. BUGS

Newline and tab are considered printing characters in filenames.

LINE

***** See Warning in Section 1.1 *****

1. NAME

line - Read one line

2. SYNOPSIS

line

3. DESCRIPTION

Line copies one line (up to a new line) from the standard input and writes it on the standard output. It returns an exit code of 1 on **EOF** and always prints at least a new line. It is often used within shell files to read from the terminal of the user.

4. SEE ALSO

sh(1)

LN

***** See Warning in Section 1.1 *****

1. NAME

ln

3. DESCRIPTION

See *cp*.

LOADMCCPASSWD

***** See Warning in Section 1.1 *****

1. NAME

loadmccpasswd - Load MCC login/password data (Software Release 5E15 and later)

2. SYNOPSIS

loadmccpasswd [-c][-t time][-s|-r]

3. DESCRIPTION

Loadmccpasswd is a UNIX process that downloads login/password data to the maintenance teletypewriter controller (MTTYC). The login/password data, read in from the */etc/dig/authfile* file, is used in conjunction with the 5E NAR MCC login Protection capability.

Loadmccpasswd recognizes the following options:

-c	Requests a conditional download. Data is only downloaded to the MTTYC when the MCC Login Protection capability is activated in the ECD.
-f	Causes the MTTY auto-lock timeout period to be set from the value *time* rather than the setting in the ECD. Valid values for *time* are 0 or 5 - 30 minutes. A value of 0 shuts off the auto-lock timeout period.
-s	Causes the auto-lock timeout capability to be suspended indefinitely. The auto-lock timeout capability is not restored until *loadmccpasswd* is executed with the restore option.
-r	Restores the auto-lock timeout capability. The auto-lock timeout capability is restored based on the timeout value setting in the equipment configuration data (ECD).

4. HEADER FILES

None

5. FILES

/etc/dig/authfile

6. SEE ALSO

None

7. DIAGNOSTICS

Loadmccpasswd returns 0 or a value of -1 in case of error. Supporting information for errors is indicated by spooler output reports.

8. LIMITATIONS

Valid for 3B21D platform only.

9. CAVEATS

Valid login/password data is downloaded to the MTTYC when the MCC Login Protection capability is activated and there are user records in the */etc/dig/authfile*. Otherwise, null data is downloaded to the MTTYC for the purpose of clearing out old login/password data.

LOGDIR

***** See Warning in Section 1.1 *****

1. NAME

logdir - Get login directory

2. SYNOPSIS

logdir [user ...]

3. DESCRIPTION

Logdir returns the login directory of a user. If no arguments are specified, the contents of the environment variable **$HOME** is listed. If the **$HOME** variable is not set (that is, unset) or set to NULL, the login directory is taken from the password file. *User* argument(s), if specified, causes the login directory for that user to be taken from the password file.

4. FILES

/etc/passwd

5. SEE ALSO

login(1), env(1)

6. DIAGNOSTICS

Returns the number of unknown users, which may be zero.

LOGIN

***** See Warning in Section 1.1 *****

1. NAME

login - Sign on

2. SYNOPSIS

login [name]

3. DESCRIPTION

Login is used at the beginning of each terminal session and allows the identification of the user to the system. It may be invoked as a command or by the system when a connection is first established. Also, *login* is invoked by the system when a previous user has terminated the initial shell by typing a *cntrl-d* to indicate an ``end-of-file.''

Login may not be invoked as a command.

Login asks for your user name (if not supplied as an argument), and, if appropriate, the password of the user. Echoing is turned off (where possible) during the typing of the password so it does not appear on the written record of the session.

At some installations, an option may be invoked that requires the user to enter a second ``machine'' password. This is prompted by the message "machine password:". Both passwords are required for a successful login.

If the login is not completed successfully within 5 tries, all later attempts fail. The TTY must be removed and restored before another login attempt is made.

After a successful login, accounting files are updated; the procedure */etc/profile* is performed; the message-of-the-day, if any, is printed; the user-ID, the group-ID, the working directory, and the command interpreter [usually *sh* (1)] is initialized; and the file **.profile** in the working directory is executed, if it exists. These specifications are found in the **/etc/passwd** file entry for the user. The name of the command interpreter is **-** followed by the last component of the pathname of the interpreter (that is, **-sh**). If this field in the password file is empty, then the default command interpreter, */bin/sh* is used.

The basic *environment* is initialized to establish the following:

HOME=*your-login-directory*

PATH=:/bin:/usr/bin

SHELL=*last-field-of-passwd-entry*

MAIL=/usr/mail/*your-login-name*

TZ=*timezone-specification*

4. FILES

/etc/utmp	Accounting
/etc/wtmp	Accounting
/usr/mail/ *your-name*	Mailbox for user *your-name*
/etc/motd	Message-of-the-day
/etc/passwd	Password file

/etc/profile	System profile
.profile	Login profile of the user.

To end a terminal session, type a *cntrl-d* to indicate an ``end-of-file.'' Note that if you are logged on to a display administration process (DAP) terminal, the user must be in the message area at the bottom of the screen before entering *cntrl-d*. Always end the terminal session and wait for the subsequent prompt to appear before powering off a terminal.

5. SEE ALSO

admin(1), mail(1), passwd(1), sh(1), su(1)

6. DIAGNOSTICS

Login incorrect if the user name or the password cannot be matched.

No shell, *cannot open password file*, or *no directory* : consult a *UNIX*® system expert.

7. ROP OUTPUT

REPT LOGIN TTYNAME x FAILED. If x is the ttyname of the master control center (MCC) or the switch control center (SCC), change ECD so that login is not invoked from ttyname x.

REPT LOGIN TTYNAME x UNABLE TO ACCESS ECD. If the ECD is unavailable, try again later.

REPT LOGIN TTYNAME x FAILED TO CREATE y. If process y cannot be executed successfully, consult a *UNIX*® system expert.

LPR

***** See Warning in Section 1.1 *****

1. NAME

lpr - Line printer spooler

2. SYNOPSIS

lpr [-s] [device]

3. DESCRIPTION

Lpr sends standard input to the local line printer [ROP (read-only printer)]. Output can be directed to other devices by either changing the device(s) associated with output class *UNIX*® system or by specifying the desired **device** as the first parameter.

Lpr assigns a job number, divides the input into parts for the spooler, adds a header to each part, sends the parts to the spooler, and prints the job number and number of parts to standard output.

If the single part option **-s** is specified, the printout is not divided into parts. This makes for a 'nicer' looking output but can cause alarms at the SCC.

4. NOTE

Many ROPs are configured to print lowercase letters as uppercase letters. If lowercase letters are desired, set switch 1 of switch block SWE8 of the ROP to the "off" state.

5. ERROR MESSAGES

lpr: cancelled non-printable character @ *X* **in input stream.**

Possible cause - attempt to print an object module. *X* is the location of the nonprintable.

lpr: aborted - can't open temp file /unixa/tmp/UA....

Possible causes - /unixa/tmp directory is missing, /unixa file system not mounted read/write, or file system full.

6. EXAMPLES

Print the output of the 'date' command on the recent change and verify terminal.

```
date | lpr ttyv
```

Print /etc/passwd on the ROP.

```
lpr </etc/passwd
```

7. FILES

/unixa/tmp/UA* File(s) passed to spooler to print.

8. BUGS

Output to unknown devices is dropped by the spooler without notice.

LS

***** See Warning in Section 1.1 *****

1. NAME

ls - List contents of directories

2. SYNOPSIS

ls [-lgtasdrucif] names

3. DESCRIPTION

For each directory named, *ls* lists the contents of that directory; for each file named, *ls* repeats its name and any other information requested. By default, the output is sorted alphabetically. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments are processed before directories and their contents. There are several options:

-l	List in long format giving mode, number of links, owner, size in bytes, and time of last modification for each file. If the file is a special file, the size field contains the major and minor device numbers rather than a size.
-g	Print group rather than owner with -l option.
-t	Sort by time of last modification (latest first) instead of by name.
-a	List all entries; in the absence of this option, entries whose names begin with a period **(.)** are *not* listed.
-s	Give size in blocks (including indirect blocks) for each entry.
-d	If argument is a directory, list only its name; often used with **-l** to get the status of a directory.
-r	Reverse the order of sort to get reverse alphabetic or oldest first, as appropriate.
-u	Use time of last access instead of last modification for sorting (with the **-t** option) and/or printing (with the **-l** option).
-c	Use time of last modification of the inode (mode, etc.) instead of last modification of the file for sorting **(-t)** and/or printing **(-l)**.
-i	For each file, print the i-number in the first column of the report.
-f	Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off **-l, -t, -s** , and **-r**, and turns on **-a**; the order is the order in which entries appear in the directory.

The mode printed under the **-l** option consists of 11 characters that are interpreted as follows:

The first character is as follows:

d	If the entry is a directory
b	If the entry is a block special file
c	If the entry is a character special file

p	If the entry is a fifo (a.k.a. ``named pipe'') special file
C	If the entry is a contiguous file
x	If the entry is a multiextent file
-	If the entry is an ordinary file.

The next nine characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the file; and the last to all others. Within each set, the three characters indicate permission to read, to write, and to execute the file as a program, respectively. For a directory, ``execute'' permission is interpreted to mean permission to search the directory for a specified file.

The permissions are indicated as follows:

r	If the file is readable
w	If the file is writable
x	If the file is executable
-	If the indicated permission is *not* granted.

The group-execute permission character is given as **s** if the file has set-group-ID mode; likewise, the user-execute permission character is given as **s** if the file has set-user-ID mode. The last character of the mode (normally **x** or **-**) is **t** if the 1000 (octal) bit of the mode is on; see *chmod* (1) for the meaning of this mode. The indications of set-ID and 1000 bit of the mode are capitalized (**S** and **T**, respectively) if the corresponding execute permission is *not* set.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks, is printed.

4. FILES

/etc/passwd	to get user IDs
/etc/group	to get group IDs.

5. SEE ALSO

chmod(1), find(1)

MAIL (5E13)

***** See Warning in Section 1.1 *****

1. NAME

mail, rmail - Send mail to users or read mail (For Software Release **5E13 Only**)

2. SYNOPSIS

mail [-pqr] [-f file]

mail persons

rmail persons

3. DESCRIPTION

Mail without arguments prints a user's mail, message-by-message, in last-in, first-out order. For each message, the user is prompted with a **?**, and a line is read from the standard input to determine the disposition of the message:

<new line>	Go on to next message.
+	Same as <new line>.
d	Delete message and go on to next message.
p	Print message again.
-	Go back to previous message.
s [*files*]	Save message in the named *files* (**mbox** is default).
w [*files*]	Save message, without its header, in the named *files* (**mbox** is default).
m [*persons*]	Mail the message to the named *persons* (yourself is default).
q	Put undeleted mail back in the *mailfile* and stop.
EOT (control-d)	Same as **q.**
x	Put all mail back in the *mailfile* unchanged and stop.
***!** command*	Escape to the shell to do *command*.
*****	Print a command summary.

The optional arguments that alter the printing of the mail are as follows:

-p	Causes all mail to be printed without prompting for disposition.
-q	Causes *mail* to terminate after interrupts. Normally, an interrupt only causes the termination of the message being printed.
-r	Causes messages to be printed in first-in, first-out order.
-f file	Causes *mail* to use *file* (for example, **mbox**) instead of the default *mailfile*.

When *persons* are named, *mail* takes the standard input up to an end-of-file (or up to a line consisting of just a **.**) and adds it to each *person's mailfile*. The message is preceded by the sender's name and a postmark. Lines that look like postmarks in the message (that is, ``From ...'') are preceded with a **>**. A *person* is usually a user name recognized by *login* (1). If a *person* being sent mail is not recognized, or if *mail* is interrupted during input, the file **dead.letter** is saved to allow editing and resending.

To denote a recipient on a remote system, prefix *person* by the system name and exclamation mark. Everything after the first exclamation mark in *persons* is interpreted by the remote system. In particular, if *persons* contains additional exclamation marks, it can denote a sequence of machines through which the message is to be sent on the way to its ultimate destination. For example, specifying **a!b!cde** as a name of the recipient causes the message to be sent to user **b!cde** on system **a.** System **a** interprets that destination as a request to send the message to user **cde** on system **b.** This might be useful, for instance, if the sending system can access system **a** but not system **b**, and system **a** has access to system **b.**

Rmail only permits the sending of mail.

When a user logs in, the presence of mail, if any, is indicated. Also, notification is made if new mail arrives while using *mail*.

4. FILES

/etc/passwd	Identifies sender and locates persons
/usr/mail/ *user*	Incoming mail for *user*; that is, the *manfile*
$HOME/mbox	Saved mail
$MAIL	Variable containing pathname of *mailfile*
/tmp/ma*	Temporary file
/usr/mail/*. lock	Lock for mail directory
dead.letter	Unmailable text

5. SEE ALSO

login(1), write(1)

6. BUGS

Race conditions sometimes result in a failure to remove a lock file.

After an interrupt, the next message may not be printed; printing may be forced by typing a **p**.

MAIL (5E14 & 5E15)

1. NAME

mail, rmail **-** Send mail to users or read mail (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

mail [+] [*-e*] [*-f file*] [*-i*] [*-p*] [*-q*] [*-r*]
mail *person1* [*person2* [...]]
rmail *person1* [*person2* [...]]
rmail *-X days person*

3. DESCRIPTION

3.1 MAIL

In the first form, *mail* reads the standard input up to an end of file or a line consisting of just a dot (.) and attempts to add it to the mail file of each user. The message is preceded by the name of the sender and a postmark. Lines that look like postmarks (that is, "From...") are prepended with a **>**. The **login** (2G) recognizes the name of the user.

If the mail file of a user is filled beyond the mail delivery limit, the sender receives a warning, and a delivery failure notice is appended to the mail file of the user, if one is not already attached.

Delivery failures, in general, result in a *dead.letter* file being created in the home directory of the user, except for the ROOT login whose *dead.letter* is created in */usr/mail*(used in 5E14 and 5E15 Software Releases).

In the second form, *mail* is used for reading mail. By default, mail is read in last-in, first-out order from the primary mail box of the user without terminating on interrupts.

The **+** and **-r** options are equivalent and cause mail messages to be printed in first-in, first-out order.

The **-f** option may be used to specify an alternate mailbox file to be read.

The **-i** option causes mail not to terminate on interrupts. This is the default action. The **-q** option causes mail to quit on interrupts, leaving the mail file unchanged.

The **-e** option causes mail to check for the existence of mail rather than actually reading it. Mail exits with a status of 0 if the user has mail and a non-zero status if not.

The **-p** option causes all mail messages to print and exits without prompting for a message disposition between messages or printing delivery limits and mail box sizes.

Normally, the user is prompted to take one of the following actions after each message:

<*newline*>	Go to the next message
+	Go to the next message
^	Go to the previous message
-	Go to the previous message
EOT (control-D)	Put unexamined mail back in the mailbox and exit
d	Delete the message and go to the next

m *[person...]*	Mail the message to each person (default is yourself)
n	Go to the next message
p	Print the message again
q	Put unexamined mail back in the mailbox and exit
s *[file...]*	Save the message in the named files (default is mbox)
w *[file...]*	Same as "*s*", but without the header
x	Exit without making any changes to the mailbox
!command	Execute the command as a shell escape
anything else	Print a command summary.

Primary mailboxes are subject to size limits which restrict the delivery of new mail. The default is 4096 bytes, but this may be overridden by adding a line with the name of the user, a space, and the desired size to the delivery limit specification file (*/usr/mail/MAXSIZE* for the 5E14 Software Release and */var/mail/MAXSIZE* for the 5E15 Software Release). The default units are bytes, but a "b", "k", or "m" may be appended to indicate blocks, kilobytes, or megabytes, respectively. The special user name "*" may be used to specify a new default.

Unless it is empty, users are informed of the delivery limit and current size when they read a primary mailbox. If some messages are deleted but the mailbox is not completely emptied, the delivery limit and new size are printed when the user exits mail.

The delivery limit specification file can also be used to prevent **mail** from using all the space in the */tmp* and */usr* for the 5E14 Software Release or */var* for the 5E15 Software Release partitions by requiring a minimum number of blocks to be maintained after mail processing (/tmp) and delivery (*/usr* for the 5E14 Software Release or */var* for the 5E15 Software Release). This is done by adding lines, to */usr/mail/MAXSIZE* in the 5E14 Software Release or */var/mail/MAXSIZE* in the 5E15 Software Release, as follows:

```
#MINMAILTMP xxx
#MINMAILDIR yyy
```

Where *xxx* is the minimum number of blocks for */tmp* and *yyy* is the minimum number of blocks for */usr* in 5E14 Software Release or */var* in 5E15 Software Release. The defaults are 500 and 1000 blocks for */tmp* and */usr* in 5E14 Software Release or */var* in 5E15 Software Release, respectively.

3.2 RMAIL

rmail is identical to the first form of **mail**. That is, **rmail** only permits the sending of mail. **uucp**(2C) uses **rmail** as a security precaution.

The **-X nnn** option causes **rmail** to remove any messages older than *nnn* days old from the mailbox of the user. Only login users root and uucp may use the *-X* option.

4. HEADER FILES

none

5. FILES

Software Release 5E14

$HOME/dead.letter	Saved letter if delivery fails for non-ROOT user
/etc/passwd	User name look-up
$HOME/mbox	Default secondary mailbox
*/tmp/ma**	Temporary files
*/usr/mail/**	Primary mail directory
/usr/mail/.lock*	Concurrency control
/usr/mail/dead.letter	Saved letter if delivery fails for ROOT user
/usr/mail/MAXSIZE	Delivery limit specification file.

Software Releases 5E15 and Later

$HOME/dead.letter	Saved letter if delivery fails for non-ROOT user
/etc/passwd	User name look-up
$HOME/mbox	Default secondary mailbox
*/tmp/ma**	Temporary files
*/var/mail/**	Primary mail directory
*/usr/mail**	Secondary mail directory if /var/mail does not exist
/var/mail/.lock*	Concurrency control
/usr/mail/.lock*	Concurrency control
/var/mail/dead.letter	Saved letter if delivery fails for ROOT user
/var/mail/MAXSIZE	Delivery limit specification file
/usr/mail/dead.letter	Saved letter if delivery fails for ROOT user
/usr/mail/MAXSIZE	Delivery limit specification file.

6. SEE ALSO

write(2G)

7. DIAGNOSTICS

Self-explanatory.

8. LIMITATIONS

Overload conditions sometimes result in a failure to remove a locked file.

MAN

***** See Warning in Section 1.1 *****

1. NAME

man - Display manual pages

2. SYNOPSIS

man [command ...] **all.man.pages**

3. DESCRIPTION

Man attempts to locate the manual page(s) for each *command* specified and then displays them on the standard output; thus, the following command displays the manual page for the **at** command:

```
man at
```

The manual pages for the **cat** and **man** commands are displayed by using the following command line.

```
man cat man
```

Some commands, such as 'true' and 'false,' share a single manual page. Thus, requesting either one results in displaying the same page.

If **all.man.pages** is specified, then the names of all manual pages are displayed.

4. FILES

/unixa/man/* Manual pages

MESG

***** See Warning in Section 1.1 *****

1. NAME

mesg - Permit or deny messages

2. SYNOPSIS

msg [n] [y]

3. DESCRIPTION

Mesg with argument **n** forbids messages via *write* (1) by revoking nonuser write permission on the user's terminal. *Mesg* with argument **y** reinstates permission. All by itself, *mesg* reports the current state without changing it.

4. FILES

/dev/tty*

5. SEE ALSO

write(1)

6. DIAGNOSTICS

Exit status is 0 if messages are receivable, 1 if not, or 2 on error.

MKDIR

***** See Warning in Section 1.1 *****

1. NAME

mkdir - Make a directory

2. SYNOPSIS

mkdir dirname ...

3. DESCRIPTION

Mkdir creates specified directories in mode 777, possibly altered by *umask* [see *sh*(1)]. Standard entries, **.** , for the directory itself, and **..** , for its parent, are made automatically.

Mkdir requires write permission in the parent directory.

4. SEE ALSO

sh(1), rm(1), umask [see sh(1)]

5. DIAGNOSTICS

Mkdir returns exit code 0 if all directories were successfully made; otherwise, it prints a diagnostic and returns nonzero.

MKDSK

***** See Warning in Section 1.1 *****

1. NAME

mkdsk - Read in disk partition(s) from tape

2. SYNOPSIS

/etc/mkdsk -i <tape> -d <vtoc> -p <list>

/etc/mkdsk -i <tape> -o <mhd>

3. DESCRIPTION

mkdsk allows one or more partitions from a Load Format (LDFT) tape to be read into disk. The first format allows selected partitions to be read into an on-line disk or disk pair. The second format reads in all partitions on the tape into a single Out-of-Service (OOS) disk.

The parameters for the first format are as follows:

<tape>	Tape device name, usually /dev/mt00 (high speed) or /dev/mt08 (low speed).
<vtoc>	vtoc filename of destination disks. /dev/vtoc for MHDs 0&1, /dev/vtoc1 for MHDs 2&3, /dev/vtoc2 for MHDs 4&5 and so on.
<list>	A filename containing the names of the partitions to be read in.

The following parameters are for the second format:

<tape>	Tape device name, usually /dev/mt00 (high speed) or /dev/mt08 (low speed).
<mhd>	Destination disk name. Examples: MHD=0, MHD=4, MHD=5.

3.1 Multi-Reel

If the tape is part of a multi-reel sequence, after each tape is read the system rewinds it and requests that the tape next be mounted. After the tape is mounted, enter the command **/etc/mkstart**. This must be done within 5 minutes or the system times out and aborts the process.

3.2 Vtoc

Be aware, the /etc/mkdsk does NOT read in if the vtoc on the tape does not agree with the vtoc on the disk. The following craft command places the correct vtoc on the MHD number specified by x:

```
EXC:ENVIR:UPROC,FN="/etc/rcvtoc",ARGS=x
```

MKNOD

***** See Warning in Section 1.1 *****

1. NAME

mknod - Build a special file

2. SYNOPSIS

/etc/mknod name [b|c|**i**|r] dcn part rid

/etc/mknod name **p**

3. DESCRIPTION

Mknod makes a directory entry and corresponding i-node for a special file. The first argument is the **name** of the entry.

The second argument identifies the special file as one of the following types:

b	Block-type (disk)
c	Character-type
i	IOP-type
r	Record-type (magnetic tape)
p	FIFO-type (named pipe).

The last three arguments are numbers specifying the device class number (**dcn**), partition (**part**), and record ID of the minor device chain table (**rid**) for the special file. The numbers may be expressed in either decimal octal notation with the latter being indicated by a leading zero. The default creation mode is 0666.

The assignment of the **dcn**, **part**, and **rid** is specific to each system and must be consistent with the equipment configuration data.

When creating a FIFO special file, the **dcn**, **part**, and **rid** are unnecessary.

MKSTART

***** See Warning in Section 1.1 *****

1. NAME

mkstart - Continues a mkdsk run after a new tape has been mounted

2. SYNOPSIS

mkstart

3. DESCRIPTION

Mkstart sends a message to *mkdsk* in order to have *mkdsk* continue its run after a new tape has been mounted. *Mkstart accepts no parameters.*

4. HEADER FILES

umsg.h, splcl.h

5. SEE ALSO

mkdsk(1)

6. DIAGNOSTICS

Processing errors are echoed to the invoking terminal. The errors that *mkstart* reports are as follows:

- A failure to get the process id of mkdsk
- Incorrect syntax of the mkstart command
- A failure to send the message to mkdsk.

MOUNT

***** See Warning in Section 1.1 *****

1. NAME

mount, umount - Mount and dismount file system

2. SYNOPSIS

/etc/mount [Special directory [-r]]

/etc/umount Special

3. DESCRIPTION

Mount announces to the system that a file system is present on the device *special.* The *directory* must already exist and becomes the name of the root of the newly mounted file system.

These commands maintain a table of mounted devices. If invoked with no arguments, *mount* prints the table.

The optional last argument indicates that the file is to be mounted read-only. Physically write-protected and magnetic tape file systems must be mounted in this way or errors occur when access times are updated, whether or not any explicit write is attempted.

Umount announces to the system that the removable file system previously mounted on device *special* is to be removed.

4. DIAGNOSTICS

Mount issues a warning if the file system to be mounted is currently mounted under another name.

Umount complains if the special file is not mounted or if it is busy. The file system is busy if it contains an open file or the working directory.

5. BUGS

Some degree of validation is done on the file system; however, it is generally unwise to mount garbage file systems.

MSNAP

***** See Warning in Section 1.1 *****

1. NAME

msnap - Memory usage snapshot program

2. SYNOPSIS

msnap -**T**type -**P** pid1 pid2 -**U** uid1 uid2 -**L**file -**S** -**A** -**F**

3. DESCRIPTION

Msnap is a user level program that collects run time information on the memory consumption of a process, shared library, the kernel, or an entire system.

Many options are provided by the tool to allow flexibility in collecting memory consumption data on a per process or system basis. Two major report types are currently supported for processes. The first is the short report (default). This report prints out the process name, utility id, process id, and tty (if a supervisor). Sharable memory consumption is provided in pages for text, data, and stack segments. Private memory consumption is provided in pages for text, data, and stack segments. In the context of this document, sharable memory is defined by the presence of a flag in the process segment list, not by whether the segment is actually shared or not. All text segments are sharable by default. For a description of the long report, see the type arguments listed in succeeding paragraphs.

3.1 Report Types

The **-T** argument controls the report type and format of data provided by the tool. The current report types are sSkKIL. The meaning of each report type follows.

s This option provides information on all active supervisor level processes.

k This option provides information on all active kernel level processes (except kernel special processes).

K This option provides memory usage information on the kernel which includes the special processes.

l This option requests that shared library information be printed. The **-L***libfile* argument must also be specified. The *libfile* keyword indicates the full or relative pathname to a file containing the shared library specifications. See the section on shared libraries for the format of this file.

S This option provides summary information on the memory consumption of the processes snapshot. Total memory used, along with summary information on sharable, actually shared, swappable, and locked swappable memory, are provided in segments, pages, and bytes.

L This option provides a *long* listing of the memory usage for the specified processes. A line is printed for each segment that the measured process owns. Segment information is output in increasing order by segment index. The segment index (SNDX), segment list flags (SFLGS), segment id (SEGID), segment descriptor flags (SDEFLAGS), process lock count (PLC), I/O lock count (IOLC), nonswap count (NSWC), active count (ACTC), number of users of the segment (NUSRS), the system name for the segment (SEG_NAME), the number of pages in the segment (PGS), the size of the last page in bytes (LSTPG), the pages in memory for supervisor processes (INM), and the swap block address for supervisor processes (SWAD) are provided. This information is collected from the processes segment list and the system segment descriptor.

3.2 Shared Libraries

Information on shared libraries can be obtained by this tool. For flexibility, a specfile (**-L***file option*) can be specified for shared libraries. The file argument is the full or relative pathname of the specfile. The specfile contains information that allows the tool to determine which processes include a particular shared library and which segments belong to the library.

Each shared library in the system has identifying attributes. The operating system allows two 32-bit words (library word 0 and 1) for library identification. Each bit in either word identifies a particular library. The msnap tool uses this means to determine which libraries a process is attached to.

The format of the specfile contains a star (*) immediately followed by the library name (limit of 16 characters). Next, the library word and bit index are specified, with each value being separated by a space or tab. Additional lines indicate the segment indexes associated with the library. The segment indexes are broken up into four groups. These groups are the Text Segments (TS), Data Segments (DS), Patch Segments (PS), and Misc Segments (MS) for any additional segments. Each segment list entry must be separated by a space character. Each segment definition must be on its own line. At least one segment type definition with at least one segment index must be specified for each library. For text and data segments, the maximum number of segments that can be specified is 8. For patch and misc segments, the maximum is 4. There is a limit of 20 library definitions in the specfile.

An example of the library specifications file entry for libc follows:

```
*UNIX_LIBC 1 3

TS 50

DS 51

PS 52
```

The library name follows the star (*) character which is the library entry separator. The "1 3" indicates library word 1, bit position 3.

The "TS 50" line indicates one text segment at segment index 50. The "DS 51" line indicates one data segment at segment index 51. The "PS 52" line indicates one patch segment at segment index 52.

3.3 Specific Processes

The **-P** and **-U** arguments allow the selection of specific processes to be measured. The **-P** argument allows for a list of up to 20 process ids (in decimal) to be specified. The **-U** argument allows for a list of up to 20 utility ids to be specified. Each utility id must be specified in hexadecimal in the following format: 0xfff, 0Xfff, or xfff.

3.4 Memory Usage Summary

The **-S** argument provides a summary of all memory used in the system. This option provides a memory usage summary based on data derived from all the system sde's. Memory consumed, free, locked, and on the swap device is printed. A limitation with this option is that a process may be created or terminated during the collection of memory data. This may cause some inconsistencies in the data provided. Therefore, this option may need to be run a few times to verify the accuracy of the information provided.

3.5 SDE Audit

The **-A** option runs a quick audit of all the system SDEs. This option runs through all process PCBs and through the system SDE segment then prints a list of segments that are allocated but not owned by any process. Note that one segment always appears. This segment is basically the low core segment and its segid is always fixed "0x1a0000". A limitation of this option is that, if a process is created or terminated during the audit, many segments may appear in the output. Under this case, the tool may need to be run several times and only the segments that repeat are the ones "really" not owned by any process.

3.6 Process Capabilities & Performance

The **-F** option provides information about each running supervisor process. The information is data gathered from each process PCB segment. The data output contains the number of capabilities owned by the process. Capabilities are used and returned by the FMGR in response to open file and change directory requests. Other information about the current state of the process is also output. In earlier versions, the DCT flags yield was output. Starting in load 1.2.25, the flags are presented as characters. Only six flags are output after load 25. They are as follows:

I	For in memory
0	For swapped out on the disk
R	For running
S	For roadblocked (sleeping)
r	For the process has run since being swapped in
N	To indicate that the process has the "NOFIT" dcte flag set.

4. FILES

msnap	-	The program normally resides in the /usr/bin directory.
libfile	-	An optional file containing shared library information.

5. SEE ALSO

Programmer's Manual Volume 1, System Architecture, Section 5, contains information that aids the user in interpreting the data provided by this tool. The user can also reference the following system header files for definition of the flags and other data provided by the program.

The following files reside in $OFC/head:

```
pcb.h

 kpcb.h

 sde.h

 pgt.h

 pge.h

 sge.h

 const.h

 dtype.h
```

va.h

6. WARNINGS

Since this program is a low level supervisor program, it may be interrupted by higher level activity while collecting data from the kernel in heavily loaded systems. Therefore, some inconsistencies may be noted for some of the data provided by the tool. Under these conditions, more than one measurement may be required to get meaningful results. However, since the program does run at the user level, it is noninterfering to system operation and can be used in live field sites.

MV

***** See Warning in Section 1.1 *****

1. NAME

mv

3. DESCRIPTION

See *cp*.

NEWGRP

***** See Warning in Section 1.1 *****

1. NAME

newgrp - Log in to a new group

2. SYNOPSIS

newgrp [-] [group]

3. DESCRIPTION

Newgrp changes the group identification of its caller, analogously to *login* (1). The same person remains logged in, and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new group ID.

Newgrp, without an argument, changes the group identification to the group in the password file; in effect, *Newgrp* changes the group identification back to the original group of the caller.

An initial **-** flag causes the environment to be changed to the one that would be expected if the user actually logged in again.

A password is demanded if the group has a password and the user does not, or if the group has a password and the user is not listed in **/etc/group** as being a member of that group.

4. FILES

/etc/group
/etc/passwd

5. SEE ALSO

login(1)

6. BUGS

There is no convenient way to enter a password into **/etc/group**. Use of group passwords is not encouraged because they encourage poor security practices. Group passwords may disappear in the future.

NEWS

***** See Warning in Section 1.1 *****

1. NAME

news - Print news items

2. SYNOPSIS

news [-a] [-n] [-s] [Items]

3. DESCRIPTION

News is used to keep the user informed of current events. By convention, these events are described by files in the directory **/usr/news** .

When invoked without arguments, *news* prints the contents of all current files in **/usr/news**, listing the most recent first, with each preceded by an appropriate header. *News* stores the "currency" time as the modification date of a file named **.news_time** in the home directory of the user (the identity of this directory is determined by the environment variable **$HOME);** only files more recent than this currency time are considered "current."

The **-a** option causes *news* to print all items, regardless of currency. In this case, the stored time is not changed.

The **-n** option causes *news* to report the names of the current items without printing their contents and without changing the stored time.

The **-s** option causes *news* to report how many current items exist, without printing their names or contents and without changing the stored time. It is useful to include such an invocation of *news* in the **.profile** file or in the **/etc/profile** of the system system's.

All other arguments are assumed to be specific news items that are to be printed.

If a *delete* is typed during the printing of a news item, printing stops and the next item is started. Another *delete*, within 1 second of the first, causes the program to terminate.

4. FILES

/etc/profile
/usr/news/*
$HOME/**.**news_time

NICE

***** See Warning in Section 1.1 *****

1. NAME

nice - Run a command at low priority

2. SYNOPSIS

nice [- increment] command [arguments]

3. DESCRIPTION

Nice executes *command* with a lower CPU scheduling priority. If given, the *increment* argument (in the range 1 through 19), is used; if the increment argument is given, an increment of 10 is assumed.

The super user may run commands with priority higher than normal by using a negative increment, for example, **--10** .

4. SEE

nohup(1)

5. DIAGNOSTICS

Nice returns the exit status of the subject command.

6. BUGS

An *increment* larger than 19 is equivalent to 19.

NOHUP

***** See Warning in Section 1.1 *****

1. NAME

nohup - Run a command immune to hang-ups and quits

2. SYNOPSIS

nohup Command [arguments]

3. DESCRIPTION

Nohup executes *command* with hang-ups and quits ignored. If output is not redirected by the user, it is sent to **nohup.out**. If **nohup.out** is not writable in the current directory, output is redirected to **$HOME/nohup.out** .

4. SEE ALSO

nice(1)

OD

***** See Warning in Section 1.1 *****

1. NAME

od - Octal dump

2. SYNOPSIS

od [-bcdosx] [file] [[+]offset[.][b]]

3. DESCRIPTION

Od dumps *a file* in one or more formats as selected by the first argument. If the first argument is missing, **-o** is default. The meanings of the format options are as follows:

-b	Interpret bytes in octal
-c	Interpret bytes in ASCII. Certain nongraphic characters appear as C escapes: null=**\0**, backspace=**\b**, form-feed=**\f**, new-line=**\n**, return= **\r**, tab=**\t**; others appear as 3-digit octal numbers.
-d	Interpret words in unsigned decimal
-o	Interpret words in octal
-s	Interpret 16-bit words in signed decimal
-x	Interpret words in hex.

The *file* argument specifies which file is to be dumped. If no file argument is specified, the standard input is used.

The offset argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as octal bytes. If "**.**" is appended, the offset is interpreted in decimal. If **b** is appended, the offset is interpreted in blocks of 512 bytes. If the file argument is omitted, the offset argument must be preceded by **+**.

Dumping continues until end-of-file.

OPENWD

***** See Warning in Section 1.1 *****

1. NAME

openwd - Allow to open (with write) block device files for mounted file systems

2. SYNOPSIS

openwd

3. DESCRIPTION

Openwd allows processes to open with write any block devices for mounted file systems. This command must precede the processes that modify file systems via the block device access method, that is, *fsdb*. This is to protect the mounted file systems from trashing due to erroneous or malicious write attempts via the block device access method to file systems. The window can be closed by using *closewd* or is closed automatically by the file manager in 20 minutes.

4. SEE ALSO

closewd(1), fsdb(1)

PARCHK

***** See Warning in Section 1.1 *****

1. NAME

parchk - Partition name check

2. SYNOPSIS

parchk [filename]

3. DESCRIPTION

Parchk reads the file *filename* or stdin, if no filename is given, for a list of partition names. Each partition name in the input stream is searched for in the on-line SG database. For each partition name matching an SG database entry, *parchk* extracts the pack number, partition number, and block size from the partition record. This information is printed on stdout in the following format:

<partition name> rt <pack number> <partition number> <block count>.

If any of the named partitions are not located, then *parchk* lists that partition name as "not found."

4. DIAGNOSTICS

Parchk fails with an error code if the on-line SG database cannot be attached or read. Appropriate error messages are printed to stderr before *parchk* exits. See the following examples:

5. EXAMPLES

```
echo "no5text" | parchk
                    or
echo "no5text" > file; parchk file
```

Either of the preceding examples results in an output similar to the following:

```
no5text rt 0 19 175000
```

Indicating that the no5text partition is on rt (pack #) 0, is partition number 19, and has 175,000 blocks allocated for it.

6. FILES

/etc/parchk

7. CAVEATS

parchk assumes that each input line contains only one partition name.

PASSWD

***** See Warning in Section 1.1 *****

1. NAME

passwd - Change login password

2. SYNOPSIS

passwd[name]

3. DESCRIPTION

Passwd changes (or installs) a password associated with the current login or a specified login *name* .

The program prompts for the old password (if any) and then for the new one (twice).

The following requirements are enforced on nonsuper users to ensure nontriviality:

(1) Each password must have at least six characters.

(2) Each password must contain at least two alphabetic characters and one numeric or special character. In this case, "alphabetic" means both uppercase and lowercase letters.

(3) The new password may not be similar to the old password nor the login name. Similar things have three or more consecutive characters in the same or reverse order. These comparisons do not distinguish between uppercase and lowercase.

Only the owner of *a name* or the super user may change a password; the owner must prove ownership by entering the old password. Only the super user can create a null password or disobey the nontriviality tests.

The password file is not changed if the new password is the same as the old password or if the password is less than 2 weeks old.

4. CONSIDERATIONS

The password 'ages' 1 week at 00:00 GMT Thursday. This occurs on Wednesday, either late afternoon or early evening within the 48 contiguous states.

Even though the super user may disobey the nontriviality tests, it is strongly recommended that they be obeyed for the security of the system.

Configurations with dial-up access are strongly urged to assign a password for the login name 'deamon'. This password is the machine password.

5. FILE

/etc/passwd

6. SEE ALSO

admin(1), login(1)

PASTE

***** See Warning in Section 1.1 *****

1. NAME

paste - Merge same lines of several files or subsequent lines of one file

2. SYNOPSIS

paste file1 file2 ...

paste -d list file1 file2 ...

paste -s [-d list] file1 file2 ...

3. DESCRIPTION

In the first two forms, *paste* concatenates corresponding lines of the given input files *file1, file2,* etc. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). It is the counterpart of *cat (1)* which concatenates vertically, that is, one file after the other. In the last preceding form, *paste* replaces the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the *tab* character or with characters from an optionally specified *list.* Output is to the standard output, so it can be used as the start of a pipe or as a filter, if **-** is used in place of a filename.

The meanings of the options are as follows:

-d	Without this option, the new-line characters of each but the last file (or last line in case of the **-s** option) are replaced by a *tab* character. This option allows replacing the *tab* character by one or more alternate characters (see the succeeding paragraphs).
list	One or more characters immediately following **-d** replace the default *tab* as the line concatenation character. The list is used circularly; that is, when exhausted, it is reused. In parallel merging (that is, no **-s** option), the lines from the last file are always terminated with a new-line character, not from the *list.* The list may contain the special escape sequences: \n (new-line), \t (tab), \ (backslash), and \0 (empty string, not a null character). Quoting may be necessary if characters have special meaning to the shell (for example, to get one backslash, use *-d*).
-s	Merge subsequent lines rather than one from each input file. Use *tab* for concatenation, unless a *list* is specified with **-d** option. Regardless of the *list,* the very last character of the file is forced to be a new-line.
-	May be used in place of any filename, to read a line from the standard input. (There is no prompting).

4. EXAMPLES

ls | paste -d" " -	List directory in one column
ls | paste - - - -	List directory in four columns
paste -s -d "\t\n" file	Combine pairs of lines into lines

5. SEE ALSO

cut(1), grep(1), pr(1)

6. DIAGNOSTICS

line too long	Output lines are restricted to 511 characters.
too many files	Except for *-s* option, no more than 12 input files may be specified.

PG

***** See Warning in Section 1.1 *****

1. NAME

pg - File perusal filter for soft-copy terminals

2. SYNOPSIS

pg [- number] [-p string] [-cefns] [+linenumber] [+/pattern] [files ...]

3. DESCRIPTION

Pg is a filter which allows the examination of *files* one screen-full at a time on a soft-copy terminal.

The filename **-** and/or NULL arguments indicate that *pg* should read from the standard input. Each screen full is followed by a prompt. If the user types a carriage return, another page is displayed; other possibilities are described in the following paragraphs.

This command is different from previous paginators, it allows the user to back up and review something that has already passed. The method for doing this is explained in the followng paragraphs.

Only the *vt100* family of terminals is supported.

The options are as follows:

-number	Specifies the size (in lines) of the window that *pg* is to use instead of the default size of 24 lines. The default number of columns is 80, which cannot be changed. (On a terminal containing 24 lines, the default window size is 23.)
-p string	Causes *pg* to use *string* as the prompt. If the prompt string contains a %d, the first occurrence of %d in the prompt is replaced by the current page number when the prompt is issued. The default prompt string is **: .**
-c	Home the cursor and clear the screen before displaying each page.
-e	Causes *pg not* to pause at the end of each file.
-f	Normally, *pg* splits lines longer than the screen width, but some sequences of characters in the text being displayed (that is, escape sequences for underlining) generate undesirable results. The **-f** option inhibits *pg* from splitting lines.
-n	Normally, commands must be terminated by a < *newline* > character. This option causes an automatic end of command as soon as a command letter is entered.
-s	Causes *pg* to print all messages and prompts in standout mode (usually inverse video).
+ linenumber	Start up at *linenumber .*
+/ pattern /	Start up at the first line containing the regular expression pattern.

The responses that may be typed when *pg* pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands which cause further perusal normally take a preceding *address*□*an optionally signed (+/-) number indicating the point from which* further text should be displayed. This .I address is interpreted in either pages or lines depending on the command. A signed *address* specifies a point relative to the current page or line, and an unsigned *address specifies an address relative to the beginning of the file.* Each command has a default address that is used if none is provided. The perusal commands and their defaults are as follows:

[+|-][num]<newline> or <blank>
: Using absolute addressing, page *num* is displayed. Using relative addressing, the page *num* forward or backward from the current page is displayed.

[+|-][num] l
: Using absolute addressing displays a page beginning at the specified line. Using relative addressing simulates scrolling the screen, forward or backward, the specified number of lines.

[+|-] d or ^D
: Simulates scrolling half a screen forward or backward.

The following perusal commands take no *address .*

. or ^L	Typing a single period causes the current page of text to be redisplayed.
$	Displays the last window full in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The regular expressions described in *ed (1)* are available. They must always be terminated by a < *newline* >*,* even if the option is specified.

i / pattern /	Searches forward for the *i th* (default *i =1)* occurrence of *pattern.* Searching begins immediately after the current page and continues to the end of the current file, without wraparound.
i ^ pattern ^ and i ? pattern ?	Searches backwards for the *i th* (default *i =1)* occurrence of *ttern .* Searching begins immediately before the current page and continues to the beginning of the current file, without wraparound.

After searching, *pg* normally displays the line found at the top of the screen. This can be modified by appending **m** or **b** to the search command to leave the line found in the middle or at the bottom of the window from now on. The suffix **t** can be used to restore the original situation.

The perusal environment can be modified with the following commands:

i n	Begin perusing the *i th* next file in the command line. The *i* is an unsigned number, default value is 1.
i p	Begin perusing the *i th* previous file in the command line. The *i* is an unsigned number, default is 1.
i w	Display another window of text. If *i* is present, set the window size to *i .*
s filename	Save the input in the named file. Only the current file being perused is saved. The white space between the **s** and *filename* is optional. This command must always be terminated by a < *newline* >*,* even if the option is specified.
h	Help by displaying an abbreviated summary of available commands.
q or Q	Quit *pg .*
! *command*	*Command* is passed to the shell, */bin/sh*. This command must always be terminated by a < *newline* >, even if the *-n* option is specified.

At any time when output is being sent to the terminal, the user can hit the quit key (normally control-ü) or the interrupt key (break). This causes *pg* to stop sending output and displays the prompt. Then the user may enter one of the preceding commands in the normal manner. Unfortunately, some output is lost when this is done, because any characters, waiting in the output queue of the terminal, are flushed when the quit signal occurs.

If the standard output is not a terminal, then *pg* acts like the cat command, except that a header is printed before each file (if there is more than one).

4. EXAMPLE

An example of *pg* usage is as follows:

```
lf -l | pg -p "(Page %d):"
```

5. NOTES

While waiting for terminal input, *pg* responds to **BREAK**, **DEL**, and **^\** by terminating execution. However, between prompts, these signals interrupt the current task of *pg* and place the user in the prompt mode. These should be used with caution when input is being read from a pipe, since an interrupt is likely to terminate the other commands in the pipeline.

6. SEE ALSO

ed(1), grep(1)

7. CAVEATS

If terminal tabs are not set every eight positions, undesirable results may occur.

When using *pg* as a filter with another command that changes the terminal I/O options, terminal settings may not be restored correctly.

PIO

***** See Warning in Section 1.1 *****

1. NAME

pio - Physical I/O

2. SYNOPSIS

pio command

3. DESCRIPTION

Pio is executed using physical I/O *to* and *from* all files where possible. If physical I/O is not possible, the system does the appropriate side-buffering.

This command is useful for doing *check*, *dd*, and other *UNIX*® system commands that make use of raw I/O.

4. EXAMPLE

```
pio dd if = /dev/ofln of = /dev/mt00 obsize = 2048 count = 512
```

5. SEE ALSO

cp(1)

PKILL

***** See Warning in Section 1.1 *****

1. NAME

pkill - Terminate a process (superuser)

2. SYNOPSIS

pkill pid

3. DESCRIPTION

Pkill creates a new supervisor process from the file in */prc/pkill*. A message with the **pid** of the process to be killed is sent to the new supervisor process. The supervisor process sends a terminate message to the process manager to take down the process designated by pid.

4. SEE ALSO

ps(1)

5. LIMITATIONS

Pkill should only be used as a last resort to terminate a process. In cases where it is necessary to forcibly terminate a copy of a shared supervisor, some allocated resources may not be freed (buffers, semaphores, and so on). Therefore, a degradation of service may occur for all users sharing that supervisor.

In short, *pkill* of a supervisor should only be done to force a core dump of a particular *UNIX*® system process before **rebooting**. If *pkill* is executed for a nonexistent or a nonkillable process ID, an error message is generated.

PR

***** See Warning in Section 1.1 *****

1. NAME

pr - Print files

2. SYNOPSIS

pr [options] [files]

3. DESCRIPTION

Pr prints the named files on the standard output. If *file* is **-** or if no files are specified, the standard input is assumed. By default, the listing is separated into pages, each headed by the page number, a date and time, and the name of the file.

By default, columns are of equal width and separated by at least one space; lines which do not fit are truncated. If the **-s** option is used, lines are not truncated and columns are separated by the separation character.

If the standard output is associated with a terminal, error messages are withheld until *pr* has completed printing.

The following *options* may appear singly or be combined in any order:

+*k*	Begin printing with page *k* (default is 1).
-*k*	Produce *k* -column output (default is 1). The options **-e** and **-i** are assumed for multicolumn output.
-a	Print multicolumn output across the page.
-m	Merge and print all files simultaneously, one per column (overrides the **-***k*, and **-a** options).
-d	Double-space the output.
-e*ck*	Expand *input* tabs to character positions k "+1, 2*" k "+1, 3*" k +1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. Tab characters in the input are expanded into the appropriate number of spaces. If *c* (any nondigit character) is given, it is treated as the input tab character (default for *c* is the tab character).
-i*ck*	In *output* , replace white space wherever possible by inserting tabs to character positions k "+1, 2*" k "+1, 3*" k +1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any nondigit character) is given, it is treated as the output tab character (default for *c* is the tab character).
-n*ck*	Provide *k* -digit line numbering (default for *k* is 5). The number occupies the first *k* +1 character positions of each column of normal output or each line of **-m** output. If *c* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a tab).
-w*k*	Set the width of a line to *k* character positions (default is 72 for equal-width multi-column output, no limit otherwise).
-o*k*	Offset each line by *k* character positions (default is 0). The number of character positions per line is the sum of the width and offset.

-l*k*	Set the length of a page to *k* lines (default is 66).
-h	Use the next argument as the header to be printed instead of the filename.
-p	Pause before beginning each page if the output is directed to a terminal (*pr* rings the bell at the terminal and wait for a carriage return).
-f	Use form-feed character for new pages (default is to use a sequence of line-feeds). Pause before beginning the first page if the standard output is associated with a terminal.
-r	Print no diagnostic reports on failure to open files.
-t	Print neither the 5-line identifying header nor the 5-line trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.
-s*c*	Separate columns by the single character *c* instead of by the appropriate number of spaces (default for *c* is a tab).

4. EXAMPLES

Print **file1** and **file2** as a double-spaced, 3-column listing headed by "file list":

pr -3dh "file list" file1 file2

Write **file1** on **file2** , expanding tabs to columns 10, 19, 28, 37, ... :

pr -e9 -t <file1 >file2

5. FILES

/dev/tty*	to suspend messages

6. SEE ALSO

cat(1)

PS

***** See Warning in Section 1.1 *****

1. NAME

ps - Report process status

2. SYNOPSIS

ps [aklxp] [s swapdev] p t tlist]

3. DESCRIPTION

Ps prints certain information about active processes. Without *options*, information is printed about processes associated with the current terminal. The output consists of a short listing containing only the terminal identifier, the process ID, and the command name. Otherwise, the information that is displayed is controlled by the selection of the following options:

a	Print information about all supervisor processes associated with terminals.
k	Print information about all kernel processes.
l	Print a long listing
x	Print information about all supervisor processes not associated with a terminal.
p	Include parent process number (cannot be used with k or l flags).
s*swapdev*	Use the file *swapdev* in place of **/dev/swap**. This is useful when examining a *corefile*.
t*tlist*	Restrict listing to data about the processes associated with the terminals given in *tlist*. *Tlist* is a list of terminal identifiers separated from one another by a space.

The column headings and the meaning of the columns in a *ps* listing of supervisor processes are as follows:

PSTAT	Status flags.
PRI	Current priority.
KTIME	Time spent in the kernel in milliseconds.
KPTIM	Time spent in kernel processes in milliseconds.
STIME	Time spent in supervisor processes in milliseconds.
TTY	Terminal associated with the process. Normally this is the same as the control channel for the process. If the control channel corresponds to a nonprinting character, a '?' is printed in this field.
PID	Process number of this process.
PID	Process number of the parent process.
UTID	Utility ID of the process.
SIZE	Size of process in bytes.

SLEEP	Bit pattern on which the process in sleeping.
CMD	Command line used to invoke the process.

The column headings and the meanings of the columns in a ps listing of kernel processes are as follows:

PSTATE	Process status flag.
PRI	Execution level.
TOUT	Real-time clock value for next time-out.
RTOUT	Interval between repetitive time-outs in milliseconds.
EVENTS	Event word.
CHAN	Control channel.
PID	Process number.
ADDR	Address of PCB segment descriptor entry.
DCT	Dispatcher control table index for the process.
UTID	Utility ID of the process.
DEVICE	Name of controller, device or process.

4. FILES

/dev/kmem

/dev/pmem

/dev/swap

/dmrt/kprc

5. SEE ALSO

kill(1)

6. NOTE

Things can change while *ps* is running; the picture it gives is only a close approximation to reality.

PST

***** See Warning in Section 1.1 *****

1. NAME

pst - Processor resource status and timing information (a performance measurement tool)

2. SYNOPSIS

pst {-Ddelay **|** -**W**} {-**T**kstafcL **|** -**P** pid pidn **|** -**U** uid1 uidn } [-**O**ks] [-**I**] [-**p**port]

3. DESCRIPTION

Pst prints information on the status of system resources and/or the activity of running processes on an active 3B20D computer system. The process is a stand-alone users-level process that collects performance information from the *UNIX*® RTR operating system kernel. It is noninterfering to application environments.

Pst can collect run time information on active processes. If requested, *pst* prints the percentage of time spent in the process and the kernel on behalf of the process (servicing ost requests). This percentage is based on the time consumed by the process over the snapshot interval. These reports print the process pid and utility id. The process name and arguments must be obtained from a ps or other listing (see **-L** flag in the following paragraphs). This is done to minimize overhead in the collection of data.

The tool works by collecting two samples of information from the address space of the kernel. The two samples or snapshots are separated by a user definable time delay. The first set of data is collected when the tool is first started or after it receives the first E_USR event (see **-D** and **-W** options). The tool then sleeps for the desired number of seconds or waits for a second E_USR event. It then collects the second set of data. After the second set of data is collected, the process calculates the differences between the two sets of data and outputs the results to stdout.

The **-T** (type) argument areas are as follows:

k	This option provides timing information for all running kernel processes. The execution level, pid, uid, and timing information is provided.
s	This option provides timing information for all active supervisor level processes. The tty, initial and current priority, pid, uid, and timing information is provided.
t	This option provides kernel timing and resource information. A global view of real-time usage is provided in histogram format for each of the 16 execution levels supported by the operating system. The consumption of major system resources is provided over the requested snapshot interval.
a	This option provides all of the t, s, and k data.
c	This option provides a bar chart of real-time consumption for each execution level.
f	This option provides a bar chart of real time availability at each execution level.
L	This option provides a *long format* report. For kernel processes, the process name is output. For supervisor processes, it includes the process name, swap and dispatch counts, along with other information. This option must be used with caution since it causes *pst* to add all pcb segments to its address space. This can cause the *pst* program to grow to a large size. Under times of extreme memory overload, the *pst* program could hang if there is not sufficient swappable memory for it to fit in the processor memory.

The **-Ddelay** parameter is used to specify the measurement interval. *delay* is the time in seconds between the two snapshots.

The **-W** option is an alternative to the -Ddelay argument. This option allows the *pst* tool to be synchronized with other activity through E_USR events. When this argument is used, the program waits for a E_USR event before taking each of the two snapshots.

The **-U** parameter allows the user to specify a list of up to 20 utility ids of processes to be snapshot. The data is the same as the **-Ts** or **-Tk** arguments, but only the processes specified are snapped. Therefore, this option cannot be used with the **-Ts** or **-Tk** parameters. All uid arguments must be in hex (prefixed with X, x, 0X, or 0x).

The **-P** parameter is the same as the **-U** parameter with the exception that process ids are specified instead of uids. The same limitation exists that the **-Ts** and **-Tk** options cannot be used with the **-P** option. These must be specified in decimal format.

The **-O** parameter specifies that OST usage counts are requested. The k and/or s flags specify that kernel and/or supervisor ost counts be reported. At least one of the **k** or **s** flags must follow the **-O** argument.

The **-I** parameter provides interrupt usage information. Both software and hardware interrupt counts are provided.

The **-pport** parameter specifies that *pst* should attach to the system port **port**. This parameter is useful when used with the **-W** argument or to guarantee that only one copy of the process is executing at any time.

4. FILES

The *pst* command may be found in /usr/bin on the 3B20D computer.

PWD

***** See Warning in Section 1.1 *****

1. NAME

pwd - Working directory name

2. SYNOPSIS

pwd

3. DESCRIPTION

Pwd prints the pathname of the working (current) directory.

4. DIAGNOSTICS

"Cannot open .." and "Read error in .." indicate possible file system trouble and should be referred to a *UNIX*® system programming counselor.

RED

***** See Warning in Section 1.1 *****

1. NAME

red

3. DESCRIPTION

See *ed*.

RM

***** See Warning in Section 1.1 *****

1. NAME

rm, rmdir - Remove files or directories

2. SYNOPSIS

rm [-fri] file ...

rmdir dir ...

3. DESCRIPTION

Rm removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. Removal of a file requires write permission in the directory of the file, but requires neither read nor write permission on the file.

If a file has no write permission and the standard input is a terminal, the permissions of the file are printed and a line is read from the standard input. If that line begins with **y**, the file is deleted; otherwise, the file remains. No questions are asked when the **-f** option is given or if the standard input is not a terminal.

If a designated file is a directory, an error comment is printed unless the optional argument **-r** has been used. In that case, *rm* recursively deletes the entire contents of the specified directory and the directory.

If the **-i** (interactive) option is in effect, *rm* asks whether to delete each file; and, under **-r** , whether to examine each directory.

Rmdir removes entries for the named directories, which must be empty.

4. DIAGNOSTICS

Generally, it is self-explanatory. It is forbidden to remove the file **..** merely to avoid the antisocial consequences of inadvertently doing something like the following:

```
rm -r .*
```

RMAIL

***** See Warning in Section 1.1 *****

1. NAME

rmail

3. DESCRIPTION

See *mail*.

RMDIR

***** See Warning in Section 1.1 *****

1. NAME

rmdir

3. DESCRIPTION

See *rm*.

RSH

***** See Warning in Section 1.1 *****

1. NAME

rsh

3. DESCRIPTION

See *sh*.

RUN

***** See Warning in Section 1.1 *****

1. NAME

run, urun - Run an environment (superuser), run a user level process

2. SYNOPSIS

run [-b] [-f]file

run [-b] file

3. DESCRIPTION

Run starts a new environment (task) either as a kernel process or a supervisor process. Although *run* can execute a user process, such use is not recommended since the process is not started with a user environment.

Urun forks and executes a *UNIX*® system user-level process.

Two optional flags are as follows:

-b	Indicates the process is to be spawned in the background and not to wait for the "death-of-a-child process. Normally, there is no wait for the death-of-a-child process.
-f	Indicates a nonzero value is to be passed in a message to the new process created from the file contents by the process manager.

4. LIMITATIONS

A process that is run using this command inherits the channel *id* of its parent (the shell). Thus, events generated for the channel of the shell (such as the *E_INT* caused by hitting the break key) are sent to the process.

SDIFF

***** See Warning in Section 1.1 *****

1. NAME

sdiff - Side-by-side difference program

2. SYNOPSIS

sdiff [options ...] file1 file2

3. DESCRIPTION

Sdiff uses the output of *diff* (1) to produce a side-by-side listing of two files indicating those lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical; a **<** in the gutter if the line only exists in *file1*; a **>** in the gutter if the line only exists in *file2*; and a **|** for lines that are different. See the following example.

```
x | y
a a
b <
c <
d d
> c
```

The following options exist with the *sdiff* command:

-w*n*	Use the next argument, *n*, as the width of the output line. The default line length is 130 characters.
-l	Only print the left side of any lines that are identical.
-s	Do not print identical lines.
-o*output*	Use the next argument, *output*, as the name of a third file that is created as a user-controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by *diff* (1), are printed; where a set of differences share a common gutter character. After printing each set of differences, *sdiff* prompts the user with a **%** and waits for one of the following user-typed commands:

l	Append the left column to the output file.
r	Append the right column to the output file.
s	Turn on silent mode; do not print identical lines.
v	Turn off silent mode.
e l	Call the editor with the left column.
e r	Call the editor with the right column.
e b	Call the editor with the concatenation of left and right.
e	Call the editor with a zero length file.
q	Exit from the program.

On exit from the editor, the resulting file is concatenated on the end of the *output* file.

4. SEE ALSO

diff(1), ed(1)

SED

***** See Warning in Section 1.1 *****

1. NAME

sed - Stream editor

2. SYNOPSIS

sed [-n][-e script] [-f sfile] [files]

3. DESCRIPTION

The *sed* command copies the named *files* (standard input default) to the standard output; they are edited according to a script of commands. The **-f** option causes the script to be taken from file *sfile*; these options accumulate. If there is just one **-e** option and no **-f** options, the flag **-e** may be omitted. The **-n** option suppresses the default output. A script consists of editing commands, one per line, of the following form:
[address [, address]] function [arguments]

In normal operation, *sed* cyclically copies a line of input into a *pattern space* (unless there is something left after a **D** command); applies in sequence all commands whose *addresses* select that pattern space; and, at the end of the script, copies the pattern space to the standard output (except under **-n**) and deletes the pattern space.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval.

An *address* is either a decimal number that counts input lines cumulatively across files; a **$** that addresses the last line of input; or a context address, that is, a /regular expression/ in the style of *ed*(1) modified as follows:

- In a context address, the construction \?regular expression?, where *?* is any character, is identical to /regular expression/. Note that in the context address \xabc \xdefx, the second **x** stands for itself so that the regular expression is **abcxdef**.
- The escape sequence \n matches a new line *embedded* in the pattern space.
- A period, **.**, matches any character except the *terminal* new line of the pattern space.
- A command line with no addresses selects every pattern space.
- A command line with one address selects each pattern space that matches the address.
- A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected; only one line is selected.) Thereafter, the process is repeated, looking again for the first address.

Editing commands can be applied only to nonselected pattern spaces by use of the negation function **!**.

In the following list of functions, the maximum number of permissible addresses for each function is indicated in parentheses.

The *text* argument consists of one or more lines; all but the last one end with \ to hide the new-line. Backslashes in text are treated like backslashes in the replacement string of an **s** command, and may be used to protect initial blanks and tabs against the stripping that is done on every script line. The *rfile* or *wfile* argument must terminate the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins. At most, there can be ten distinct *wfile* arguments.

(1)**a**	
text	Append. Place *text* on the output before reading the next input line.
(2)**b**label	Branch to the **:** command bearing the *label*. If *label* is empty, branch to the end of the script.
(2)**c**	
text	Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle.
(2)**d**	Delete the pattern space. Start the next cycle.
(2)**D**	Delete the initial segment of the pattern space through the first new line. Start the next cycle.
(2)**g**	Replace the contents of the pattern space by the contents of the hold space.
(2)**G**	Append the contents of the hold space to the pattern space.
(2)**h**	Replace the contents of the hold space by the contents of the pattern space.
(2)**H**	Append the contents of the pattern space to the hold space.
(1) **i**	Insert.
text	Place *text* on the standard output.
(2)**l**	List the pattern space on the standard output in an unambiguous form. Nonprinting characters are spelled in 2-digit ASCII and long lines are folded.
(2)**n**	Copy the pattern space to the standard output. Replace the pattern space with the next line of input.
(2)**N**	Append the next line of input to the pattern space with an embedded new line. (The current line number changes.)
(2)**p**	Print. Copy the pattern space to the standard output.
(2)**P**	Copy the initial segment of the pattern space through the first new line to the standard output.
(1)**q**	Quit. Branch to the end of the script. Do not start a new cycle.
(2)**r** *rfile*	Read the contents of *rfile*. Place them on the output before reading the next input line.

(2)**s** */regular expression/replacement/flags* Substitute the *replacement* string for instances of the *regular expression* in the pattern space. Any character may be used instead of ***/*** . For a fuller description, see *ed* (1). *Flags* is zero or more of the following:

g	Global. Substitute for all nonoverlapping instances of the *regular expression* rather than just the first one.
p	Print the pattern space if a replacement was made.
w *wfile*	Write. Append the pattern space to *wfile* if a replacement was made.

(2)**t** *label*	Test. Branch to the **:** command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a**t**. If *label* is empty, branch to the end of the script.
(2)**w** *wfile*	Write. Append the pattern space to *wfile* .
(2)**x**	Exchange the contents of the pattern and hold spaces.
(2)**y** */string1/string2/*	Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.
(2)**!** *function*	Don't. Apply the *function* (or group, if *function* is {}) only to lines *not* selected by the address(es).
(0)**:** *label*	This command does nothing; it bears a *label* for which **b** and **t** commands to branch.
(1)**=**	Place the current line number on the standard output as a line.
(2)**{**	Execute the following commands through a matching **}** only when the pattern space is selected.
(0)	An empty command is ignored.

4. SEE ALSO

ed(1), grep(1)

SH

***** See Warning in Section 1.1 *****

1. NAME

sh, rsh - Shell, the standard/restricted command programming language

2. SYNOPSIS

sh [-ceinrstuvx] [args]

rsh [-ceinrstuvx] [args]

3. DESCRIPTION

Sh is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. *Rsh* is a restricted version of the standard command interpreter. *Sh* is a command programming language that executes commands read from a terminal or a file. See *Invocation* for the meaning of arguments to the shell.

3.1 Commands

A *simple-command* is a sequence of nonblank *words* separated by *blanks* (a *blank* is a tab or a space). The first word specifies the name of the command to be executed. Except as specified in succeeding paragraphs, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0. The *value* of a simple-command is its exit status if it terminates normally, or (octal) 200+*status* if it terminates abnormally [see *kill* (1) for a list of status values].

A *pipeline* is a sequence of one or more *commands* separated by **^** (or by a |). The standard output of each command but the last is connected by a pipe to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate.

A *list* is a sequence of one or more pipelines separated by **;, &, &&**, or **||** , and optionally terminated by **;** or **&**. Of these four symbols, **;** and **&** have equal precedence, which is lower than that of **&&** and **||** . The symbols **&&** and **||** also have equal precedence. A semicolon (**;**) causes sequential execution of the preceding pipeline; an ampersand (**&**) causes asynchronous execution of the preceding pipeline (that is, the shell does *not* wait for that pipeline to finish). The symbol **&&** (**||**) causes the *list* following it to be executed only if the preceding pipeline returns a zero (nonzero) exit status. An arbitrary number of new lines may appear in a *list*, instead of semicolons, to delimit commands.

A *command* is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

for *name* **in** *word* ... **do** *list* **done**

Each time a **for** command is executed, *name* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, then the **for** command executes the **do** *list* once for each positional parameter that is set (see *Parameter Substitution* below). Execution ends when there are no more words in the list.

case *word* **in** *pattern* | *pattern* ...) *list* ;; ... **esac**

A **case** command executes the *list* associated with the first *pattern* that matches *word* . The form of the patterns is the same as that used for file-name generation (see *Filename Generation*).

if *list* **then** *list* **elif** *list* **then** *list* ... **else** *list* **fi**

The *list* following **if** is executed; and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed; and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else** *list* is executed. If no **else** *list* or **then** *list* is executed, then the **if** command returns a zero exit status.

while *list* **do** *list* **done**

A **while** command repeatedly executes the **while** *list*; and, if the exit status of the last command in the list is zero, executes the **do** *list*. Otherwise, the loop terminates. If no commands in the **do** *list* are executed, then the **while** command returns a zero exit status; **until** may be used in place of **while** to negate the loop termination test.

(*list***)**

Execute *list* in a subshell.

{*list***;}**

list is simply executed.

The following reserved words are only recognized as the first word of the command and then not quoted:

```
if n if then else elif fi case esac for while until do done { }
```

3.2 Comments

A word beginning with # causes that word and all the following characters up to a new line to be ignored.

3.3 Aliasing

The first word of each command is replaced by the text of an alias, if an alias for this word has been defined. The replacement string can contain any character excluding ";". Aliases can be used to redefine special built-in commands but cannot be used to redefine the reserved words. Aliases can be created, modified, and listed with the alias command and can be removed with the unalias command.

Aliasing is performed when scripts are read, not while they are executed. Therefore, for an alias to take effect the alias definition command has to be executed before the command which references the alias is read.

Aliases are frequently used as a shorthand for full pathnames or to customize the behavior of other shell commands.

3.4 Command Substitution

The standard output from a command enclosed in a pair of grave accents (``) may be used as part or all of a word; trailing new lines are removed.

3.5 Parameter Substitution

The character **$** is used to introduce substitutable *parameters*. Positional parameters may be assigned values by **set**. Variables may be set by writing the following:

```
name = value
  name = value
```

Pattern-matching is not performed on *value* .

${*parameter* **}**

A *parameter* is a sequence of letters, digits, or underscores (a *name*); a digit, or any of the characters ***, @, #,?, -, $**, and **!**. The value, if any, of the parameter is substituted. The braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. A *name* must begin with a letter or underscore. If *parameter* is a digit, then it is a positional parameter. If *parameter* is ***** or @ , then all the positional parameters, starting with **$1**, are substituted (separated by spaces). Parameter **$0** is set from argument zero when the shell is invoked.

${*parameter* **:-***word* **}**
If *parameter* is set and is nonnull, then substitute its value; otherwise, substitute *word*.

${*parameter* **:=***word* **}**
If *parameter* is not set or is null, then set it to *word*; the value of the parameter is then substituted. Positional parameters may not be assigned to in this way.

${*parameter* **:?***word* **}**
If *parameter* is set and is nonnull, then substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, then the message "parameter null or not set" is printed.

${*parameter* **:+***word* **}**
If *parameter* is set and is nonnull, then substitute *word*; otherwise, substitute nothing.

In the preceding, *word* is not evaluated unless it is to be used as the substituted string; therefore, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-pwd}
```

If the colon (**:**) is omitted from the preceding expressions, then the shell only checks whether *parameter* is set or not.

The following parameters are automatically set by the shell:

#	The number of positional parameters in decimal.
-	Flags supplied to the shell on invocation or by the **set** command.
?	The decimal value returned by the last synchronously executed command.
$	The process number of this shell.
!	The process number of the last background command invoked.

The following parameters are used by the shell:

HOME	The default argument (home directory) for the *cd* command.
PATH	The search path for commands (see *Execution*). The user may not change **PATH** if executing under *rsh*.
MAIL	If this variable is set to the name of a mail file, then the shell informs the user of the arrival of mail in the specified file.
PS1	Primary prompt string, by default **" $"**.
PS2	Secondary prompt string, by default **" > "**.
IFS	Internal field separators, normally **space, tab**, and **new-line**.
ENV	If this parameter is set, the value is used as the pathname to the script that is executed when the shell is invoked. (See *invocation*.) This file is typically used for alias definitions.

The shell gives default values to **PATH**, **PS1**, **PS2**, and **IFS**, while **HOME** and **MAIL** are not set at all by the shell [although **HOME** *is* set by *login* (1)].

3.6 Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in **IFS**) and split into distinct arguments where such characters are found. Explicit null arguments ("" or ' ') are retained. Implicit null arguments (those resulting from *parameters* that have no values) are removed.

3.7 Filename Generation

Following substitution, each command *word* is scanned for the characters *****, **?**, and **[**. If one of these characters appears, then the word is regarded as a *pattern*. The word is replaced with alphabetically sorted filenames that match the pattern. If no filename is found that matches the pattern, then the word is left unchanged. The character **.** at the start of a filename or immediately following a **/** , as well as the character / itself, must be matched explicitly.

*****	Matches any string, including the null string.
?	Matches any single character.
[...]	Matches any one of the enclosed characters. A pair of characters separated by **-** matches any character lexically between the pair, inclusive.

3.8 Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

; & ( ) | ^ < > new-line space tab

A character may be *quoted* (that is, made to stand for itself) by preceding it with a \. The pair **\\new-line** is ignored. All characters enclosed between a pair of single quote marks (''), except a single quote, are quoted.

Inside double quote marks (**""**), parameter and command substitution occurs; \ quotes the characters \ , , ", and **$**. "$*" is equivalent to **"$1 $2** ...**"**; whereas, "$@" is equivalent to "$1" "$2" ....

3.9 Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. If, at any time, a new line is typed and further input is needed to complete a command, then the secondary prompt (that is, the value of **PS2**) is issued.

3.10 Input/Output

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. The following may appear anywhere in a simple-command or may precede or follow a *command* and are *not* passed on to the invoked command. Substitution occurs before *word* or *digit* is used:

<word	Use file *word* as standard input (file descriptor 0).
>word	Use file *word* as standard output (file descriptor 1). If the file does not exist then it is created; otherwise, it is truncated to zero length.
>>word	Use file *word* as standard output. If the file exists, then output is appended to it (by first

seeking to the end-of-file); otherwise, the file is created.

<<-word The shell input is read up to a line that is the same as *word* or to an end-of-file. The resulting document becomes the standard input. If any character of *word* is quoted, then no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, (unescaped) **\new-line** is ignored, and **** must be used to quote the characters ****, **$**, and the first character of *word*. If **-** is appended to **<<**, then all leading tabs are stripped from *word* and from the document.

<&digit The standard input is duplicated from file descriptor *digit*. Similarly for the standard output using **>**.

<&- The standard input is closed. Similarly for the standard output using **>**.

If one of the aforementioned is preceded by a digit, then the file descriptor created is that specified by the digit (instead of the default 0 or 1). For example, the following command creates file descriptor 2 that is a duplicate of file descriptor 1.:

```
... 2 >&1
```

If a command is followed by **&**, then the default standard input for the command is the empty file **/dev/null**. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Redirection of output is not allowed in the restricted shell.

3.11 Environment

The *environment* is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. Executed commands inherit the same environment. If the user modifies the values of these parameters or creates new ones, none of these affects the environment unless the **export** command is used to bind the parameter of the shell to the environment. The environment, seen by any executed command, is thus composed of any unmodified name-value pairs originally inherited by the shell, plus any modifications or additions; all of which must be noted in **export** commands.

The environment for any *simple-command* may be augmented by prefixing it with one or more assignments to parameters, thus the following are equivalent (as far as the shown execution of *cmd* is concerned) :

```
TERM=450 cmd args    and
  (export TERM; TERM=450; cmd args)
```

If the **-k** flag is set, *all* keyword arguments are placed in the environment, even if they occur after the command name. The following first prints **a=b c** and then **c** :

```
echo a=b c
  set -k
  echo a=b c
```

3.12 Signals

The INTERRUPT and QUIT signals for an invoked command are ignored if the command is followed by **&**; otherwise, signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the **trap** command).

3.13 Execution

Each time a command is executed, the aforementioned substitutions are carried out. Except for the *Special Commands* listed in the succeeding paragraph, a new process is created and an attempt is made to execute the command.

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (**:**). The default path is **:/bin:/usr/bin** (specifying the current directory, **/bin**, and **/usr/bin**, in that order). Note that the current directory is specified by a null pathname, which can appear immediately after the equal sign or between the colon delimiters anywhere in the path list. If the command name contains a /, then the search path is not used; such commands are not executed by the restricted shell. Otherwise, each directory in the path is searched for an executable file. If the file has execute permission but is not an **a.out** file, it is assumed to be a file containing shell commands. A subshell (that is, a separate process) is spawned to read it. A parenthesized command is also executed in a subshell.

3.14 Special Commands

The following commands are executed in the shell process and, except as specified, no input/output redirection is permitted for such commands:

: No effect; the command does nothing. A zero exit code is returned.

. *file* Read and execute commands from *file* and return. The search path specified by **PATH** is used to find the directory containing *file*.

alias *[name[=value] ...]*

Alias with no arguments prints the list of aliases in the form name=value on standard output. If a name is specified with no value, the name and value of the alias is printed. If both a name and value are specified, an alias is defined. There are some size limitations on the alias name and value. **name** can be no more than 40 characters. **value** can contain up to 10 space separated strings, each of which can be up to 40 characters. A value containing spaces or other shell special characters must be quoted.

break *n*

Exit from the enclosing **for** or **while** loop, if any. If *n* is specified, then break *n* levels.

continue *n*

Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified, then resume at the *n*th enclosing loop.

cd *arg*

Change the current directory to *arg*. The shell parameter **HOME** is the default *arg*. The default path is **<null>** (specifying the current directory). Note, the current directory is specified by a null pathname, which can appear immediately after the equal sign or between the colon delimiters anywhere in the path list. If *arg* begins with a /, then the search path is not used. Otherwise, each directory in the path is searched for *arg*. The *cd* command may not be executed by *rsh*.

eval *arg* ...

The arguments are read as input to the shell and the resulting command(s) executed.

exec *arg* ...

The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments may appear and, if no other arguments are given, cause the shell input/output to be modified.

exit *n*

Causes a shell to exit with the exit status specified by *n*. If *n* is omitted, then the exit status is that of the last command executed (an end-of-file will also cause the shell to exit). The exit command is ignored in interactive (connected to a TTY) shells.

export *name* ...

The given *name*s are marked for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, then a list of all names that are exported in this shell is printed.

newgrp *arg* ...

Equivalent to **exec newgrp** *arg* ....

read *name* ...

One line is read from the standard input and the first word is assigned to the first *name*, the second word to the second *name*; etc., with leftover words assigned to the last *name*. The return code is 0 unless an end-of-file is encountered.

readonly *name* ...

The given *name*s are marked *read only* and the values of the these *name*s may not be changed by subsequent assignment. If no arguments are given, then a list of all *read only* names is printed.

set -ekntuvx *arg* ...

-e	Exit immediately if a command exits with a nonzero exit status.
-k	All keyword arguments are placed in the environment for a command, not just those that precede the command name.
-n	Read commands but do not execute them.
-t	Exit after reading and executing one command.
-u	Treat unset variables as an error when substituting.
-v	Print shell input lines as they are read.
-x	Print commands and their arguments as they are executed.
--	Do not change any of the flags; useful in setting **$1** to **-**.

Using **+** rather than **-** causes these flags to be turned off. Also, these flags can be used upon invocation of the shell. The current set of flags may be found in **$-**. The remaining arguments are positional parameters and are assigned, in order, to **$1**, **$2**, ... If no arguments are given, then the values of all names are printed.

shift *n* The positional parameters from **$n+1** ... are renamed **$1** .... If *n* is not given, it is assumed to be 1.

test **test** expr **[** expr **]**

Test evaluates the expression *expr* and, if its value is true, returns a zero (true) exit status; otherwise, a nonzero (false) exit status is returned; *test* also returns a nonzero exit status if there are no arguments. The following primitives are used to construct *expr*:

-r *file*	True if *file* exists and is readable.
-w *file*	True if *file* exists and is writable.
-x *file*	True if *file* exists and is executable.
-f *file*	True if *file* exists and is a regular file.

-d *file*	True if *file* exists and is a directory.
-c *file*	True if *file* exists and is a character special file.
-b *file*	True if *file* exists and is a block special file.
-p *file*	True if *file* exists and is a named pipe (fifo).
-u *file*	True if *file* exists and its set-user-ID bit is set.
-g *file*	True if *file* exists and its set-group-ID bit is set.
-k *file*	True if *file* exists and its sticky bit is set.
-s *file*	True if *file* exists and has a size greater than zero.
-t [*fildes*]	True if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.
-z *s1*	True if the length of string *s1* is zero.
-n *s1*	True if the length of the string *s1* is nonzero.
s1 = s2	True if strings *s1* and *s2* are identical.
s1 != s2	True if strings *s1* and *s2* are *not* identical.
s1	True if *s1* is *not* the null string.
n1 -eq n2	True if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **-ne, -gt, -ge, -lt**, and **-le** may be used in place of **-eq**.

These primaries may be combined with the following operators:

!	Unary negation operator.
-a	Binary *and* operator.
-o	Binary *or* operator **(-a** has higher precedence than **-o**).
(expr **)**	Parentheses for grouping.

Notice that all the operators and flags are separate arguments to *test*. Notice also that parentheses are meaningful to the shell. Therefore, the parentheses must be escaped.

times

Print the accumulated user and system times for processes run from the shell.

trap *arg n* ...

The *arg* command is to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) produces an error. If *arg* is absent, then all trap(s) *n* are reset to their original values. If *arg* is the null string, then this signal is ignored by the shell and by the commands it invokes. If *n* is 0, then the command *arg* is executed on exit from the shell. The **trap** command, with no arguments, prints a list of commands associated with each signal number.

unalias *[name ...*

The name given is removed from the alias list.

umack *nnn*

The user file-creation mask is set to *nnn*. If *nnn* is omitted, the current value of the mask is printed.

wait *n*

Wait for the specified process and report its termination status. If *n* is not given, then all currently active child processes are waited for and the return code is zero.

3.15 Invocation

If the shell is invoked through *login* and the first character of argument zero is **-** , commands are initially read from /etc/profile and then from $HOME/.profile (if such files exist). Next, commands are read from the file specified by the environment parameter **ENV** if the file exists. Thereafter, commands are read as described following this paragraph, which is also the case when the shell is invoked as /bin/sh. These flags are interpreted by the shell on invocation only. Note that unless the **-c** or **-s** flag is specified, the first argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file.

-c *string*	If the **-c** flag is present, then commands are read from *string*.
-s	If the **-s** flag is present or if no arguments remain, then commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output is written to file descriptor 2.
-i	If the **-i** flag is present or if the shell input and output are attached to a terminal, then this shell is *interactive*. In this case, TERMINATE is ignored (so that **kill 0** does not kill an interactive shell) and INTERRUPT is caught and ignored (so that **wait** is interruptible). In all cases, QUIT is ignored by the shell.
-r	If the **-r** flag is present, the shell is a restricted shell.

The remaining flags and arguments are described under the preceding **set** command.

3.16 Rsh Only

The *rsh* command is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of *rsh* are identical to those of *sh*, except that the following are disallowed:

- Changing directory (**cd**)
- Setting the value of **$PATH**
- Specifying path or command names containing **/.**
- Redirecting output (**>** and **>>**)

The preceding restrictions are enforced after **.profile** is interpreted.

When a command to be executed is found to be a shell procedure, *rsh* invokes *sh* to execute it. Thus, it is possible to provide to the end-user with shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the **.profile** has complete control over user actions by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (that is, /usr/rbin) that can be safely invoked by *rsh*. Some systems also provide a restricted editor *red*.

4. EXIT STATUS

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. If the shell is being used noninteractively, then execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed (see also the preceding **exit** command).

5. NOTE

See the stty(1) command about problems entering a **|** character.

6. FILES

/etc/profile

$HOME/**.**profile

/tmp/sh*

/dev/null

7. SEE ALSO

env(1), login(1), stty(1)

8. BUGS

The command **readonly** (without arguments) produces the same output as the command **export**. If **<<** is used to provide standard input to an asynchronous process invoked by **&**, the shell gets mixed up about naming the input document; a garbage file **/tmp/sh*** is created and the shell complains about not being able to find that file by another name.

SLEEP

***** See Warning in Section 1.1 *****

1. NAME

sleep - Suspend execution for an interval

2. SYNOPSIS

sleep time

3. DESCRIPTION

Sleep suspends execution for *time* seconds. It is used to execute a command after a certain amount of time as in the following example:

```
(sleep 105; command)&
```

An alternative is to execute a command every so often, as in the following options:

- while true
- do
- command
- sleep 37
- done

4. BUGS

Time must be less than 65,536 seconds.

SORT

***** See Warning in Section 1.1 *****

1. NAME

sort - Sort and/or merge files

2. SYNOPSIS

sort [-cmubdfinrtx] [+pos1 [-pos2]] ... [-o output] [names]

3. DESCRIPTION

Sort sorts lines of the named files together and writes the result on the standard output. The name **-** means the standard input. If no input files are named, the standard input is sorted.

The default sort key is an entire line. Default ordering is lexicographic by bytes in machine collating sequence. The ordering is affected globally by the following options, one or more of which may appear.

b	Ignore leading blanks (spaces and tabs) in field comparisons.
d	``Dictionary'' order: only letters, digits, and blanks are significant in comparisons.
f	Fold uppercase letters onto lowercase.
i	Ignore characters outside the ASCII range 040-0176 in nonnumeric comparisons.
n	An initial numeric string, consisting of optional blanks, optional minus sign, and zero or more digits with optional decimal point, is sorted by arithmetic value. Option **n** implies option **b**.
r	Reverse the sense of comparisons.
t*x*	``Tab character'' separating fields is *x*.

The notation + *"pos1" - pos2* restricts a sort key to a field beginning at *pos1* and ending just before *pos2*. *Pos1* and *pos2* each have the form *m.n*, optionally followed by one or more of the flags **bdfinr**, where *m* tells a number of fields to skip from the beginning of the line and *n* tells a number of characters to skip further. If any flags are present, they override all the global ordering options for this key. If the **b** option is in effect, *n* is counted from the first nonblank in the field; **b** is attached independently to *pos2*. A missing **.n** means **.0**; a missing **- pos2** means the end of the line. Under the **-t***x* option, fields are strings separated by *x*; otherwise, fields are nonempty, nonblank strings separated by blanks.

When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant.

These option arguments are also understood:

c	Check that the input file is sorted according to the ordering rules; give no output unless the file is out of sort.
m	Merge only; the input files are already sorted.
u	Suppress all but one in each set of equal lines. Ignored bytes and bytes outside keys do not participate in this comparison.
o	The next argument is the name of an output file to use instead of the standard output. This

file may be the same as one of the inputs.

4. EXAMPLES

Print in alphabetical order all the unique spellings in a list of words (capitalized words differ from uncapitalized):

```
sort -u +0f +0 list
```

Print the password file *(/etc/passwd)* sorted by user ID (the third colon-separated field):

```
sort -t: +2n /etc/passwd
```

Print the first instance of each month in an already sorted file of (month-day) entries (the options **-um** with just one input file make the choice of a unique representative from a set of equal lines predictable):

```
sort -um +0 -1 dates
```

5. FILES

/usr/tmp/stm???

6. DIAGNOSTICS

Comments and exits with nonzero status for various trouble conditions and for disorder discovered under option **-c**.

7. BUGS

Very long lines are silently truncated.

SPLIT

***** See Warning in Section 1.1 *****

1. NAME

split - Split a file into pieces

2. SYNOPSIS

split [-n] [file [prefix]]

3. DESCRIPTION

Split reads *file* and writes it in *n* -line pieces (default 1000 lines) onto a set of output files. The name of the first output file is *prefix* with **aa** appended, and so on lexicographically, up to **zz** (a maximum of 676 files). *Name* cannot be longer than 12 characters. If no output prefix is given, **x** is default.

If no input file is given or if **-** is given instead, the standard input file is used.

STTY

***** See Warning in Section 1.1 *****

1. NAME

stty - Set the options for a terminal

2. SYNOPSIS

stty [-g] [options]

3. DESCRIPTION

Stty sets certain terminal I/O options for the device that is the current standard input; without arguments, it reports the settings of certain options; with the **-g** option, it reports current settings in a form that can be used as an argument to another *stty* command. Note that many combinations of options make no sense, but no sanity checking is performed. The options are selected from the following:

even (-even)
Allow (dissallow) even parity.

odd (-odd)
Allow (dissallow) odd parity.

0
Hang up phone line immediately.

50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 exta extb
Set terminal baud rate to the number given, if possible. (All speeds are not supported by all hardware interfaces.)

hup (-hup)
Hang up (do not hang up) *DATAPHONE*® service connection on last close.

nl (-nl)
Allow only new-line (allow carriage return) to end input lines.

lcase (-lcase)
Map (do not map) uppercase alphabetics to lowercase.

LCASE (-LCASE)
Map (do not map) uppercase alphabetics to lowercase.

cr0 cr1 cr2 cr3
Select delay for carriage returns (larger numbers are slower).

nl0 nl1 nl2 nl3
Select delay for line-feeds (larger numbers are slower).

tab0 tab1
Select delay for horizontal tabs (larger numbers are slower).

bs0 bs1
Select delay for backspaces (larger numbers are slower).

ff0 ff1
Select delay for form-feeds (larger numbers are slower).

cooked or -raw (raw)
Enable (disable) canonical input (ERASE and KILL processing).

echo (-echo)
Echo back (do not echo back) every character typed.

lfkc (-lfkc)
Echo (do not echo) LF after kill character.

erase *c*
Set erase character to c.

kill *c*
Set kill line character to c.

tabs (-tabs or **tab3)**
Preserve (expand to spaces) tabs when printing.

ek
Reset ERASE and KILL characters back to normal # and @.

sane
Resets all modes to some reasonable values.

term
Set all modes suitable for the terminal type *term*, where *term* is one of **tty33, tty37, vt05, tn300, ti700**, or **tek** .

3.1 TN83 TN983

Terminals connected to TN83, TN983, and certain other specialized asynchronous interface controllers operate in an unusual manner. Certain ones have a line speed that is not selectable. Also, the START/STOP (^S/^Q) characters have only a temporary effect. ^X/^Z can be used on these terminals to START/STOP the output.

3.2 ECD

Some terminal ports may be configured within the ECD such that backslash (\) does not work and the pipe character (|) hangs the terminal. This can be corrected by using RCV to change form 'ttopt'. Change, on option_name 'UNIX', item 47 (escape) to xff and item 48 (escape_char) to x5c. For this to take effect, the terminal port (TTY) must be removed (RMV) and restored (RST) .

SU

***** See Warning in Section 1.1 *****

1. NAME

su - Become super user or another user

2. SYNOPSIS

su [-] [name [arg ...]]

3. DESCRIPTION

Su allows one to become another user without logging off. The default user *name* is **root** (that is, super user).

To use *su*, the appropriate password must be supplied (unless one is already super user). If the password is correct, *su* executes a new shell with the user ID set to that of the specified user. To restore normal user ID privileges, type an **EOF** to the new shell.

Any additional arguments are passed to the shell, permitting the super user to run shell procedures with restricted privileges (an *arg* of the form **-c** *string* executes *string* via the shell). When additional arguments are passed, /bin/sh is always used. When no additional arguments are passed, *su* uses the shell specified in the password file.

An initial **-** flag causes the environment to be changed to the one that would be expected if the user actually logged in again. This is done by invoking the shell with an *arg0* of **-su** causing the **.profile** in the home directory of the new user ID to be executed. Otherwise, the environment is passed along with the possible exception of **$PATH** which is set to **/bin:/etc:/usr/bin** for root. Note that the **.profile** can check *arg0* for **-sh** or **-su** to determine how it was invoked.

4. FILES

/etc/passwd	System password file
$HOME/ . profile	User profile

5. SEE ALSO

env(1), login(1), sh(1)

SUM

***** See Warning in Section 1.1 *****

1. NAME

sum - Print checksum and block count of a file

2. SYNOPSIS

sum [**-r**] file

3. DESCRIPTION

Sum calculates and prints a 16-bit checksum for the named file and also prints the number of blocks in the file. It is typically used to look for bad spots or to validate a file communicated over some transmission line. The option **-r** causes an alternate algorithm to be used in computing the checksum.

4. SEE ALSO

wc(1)

5. DIAGNOSTICS

"Read error'' is indistinguishable from end-of-file on most devices; check the block count.

SYNC

***** See Warning in Section 1.1 *****

1. NAME

sync - Update the super block

2. SYNOPSIS

sync

3. DESCRIPTION

Sync executes the *sync* system primitive. If the system is to be stopped, *sync* must be called to insure file system integrity. It flushes all previously unwritten system buffers out to disk assuring that all file modifications up to that point are saved.

TAIL

***** See Warning in Section 1.1 *****

1. NAME

tail - Deliver the last part of a file

2. SYNOPSIS

tail [+- [number][ibc [f]]] [file]

3. DESCRIPTION

Tail copies the named file to the standard output beginning at a designated place. If no file is named, the standard input is used.

Copying begins at distance **+** *number* from the beginning or **-** *number* from the end of the input (if *number* is null, the value 10 is assumed). *Number* is counted in units of lines, blocks, or characters, according to the appended option **l, b**, or **c**. When no units are specified, counting is by lines.

If the input file is not a pipe and the **-f** (``follow'') option is used, the program does not terminate after the line of the input file has been copied but enters an endless loop; the program sleeps for a second then attempts to read and copy further records from the input file. Thus, it may be used to monitor the growth of a file that is being written by some other process. For example, the following command prints the last ten lines of the file **fred** then prints any lines that are appended to **fred** between the time *tail* is initiated and killed.

```
tail -f fred
```

The following command is another example:

```
tail -15cf fred
```

This command prints the last 15 characters of the file **fred** then prints any lines that are appended to **fred** between the time *tail* is initiated and killed.

4. SEE ALSO

dd(1)

5. BUGS

Tails relative to the end of the file are treasured up in a buffer and are limited in length. Various kinds of irregular behavior may happen with character special files.

TEE

***** See Warning in Section 1.1 *****

1. NAME

tee - Pipe fitting

2. SYNOPSIS

tee [-i] [-a] [file] ...

3. DESCRIPTION

Tee transcribes the standard input to the standard output and makes copies in the *files* . The **-i** option ignores interrupts and the **-a** option causes the output to be appended to the *files* rather than overwriting them.

TIME

***** See Warning in Section 1.1 *****

1. NAME

time - Time a command

2. SYNOPSIS

time command

3. DESCRIPTION

After the command process is complete, *time* prints the elapsed time during the command, the time spent in the system, and the time spent in executing the command. Times are reported in seconds.

The execution time can depend on the type of memory in which the program is stored; the user time in the Module Operating System (MOS) is often half the user time in the core.

The times are printed on standard error.

TIMER

***** See Warning in Section 1.1 *****

1. NAME

timer - Find processes with large central processing unit (CPU) usage

2. SYNOPSIS

timer seconds

3. DESCRIPTION

Due to unexpected events, sometimes a process starts using large amounts of CPU time. When this occurs, the *UNIX*® system level processes within the AM (3B20D), such as recent change, may slow down to an unacceptable speed. The **timer(1)** command aids in finding these large CPU-using processes.

The timer(1) operates by doing two ps(1) commands several minutes apart. **Timer(1)** then calculates the CPU time used by each process between the two ps(1) commands and prints a message showing the 12 highest CPU using processes.

The user specifies the time between the ps(1) commands, in seconds, as the argument to the command. Valid times are from 30 to 3600 seconds.

To enter this command from the craft shell with a 120-second interval, enter the following command for MML:

```
EXC:ENVIR:UPROC,FN="/bin/timer",ARGS=120;
```

To execute the same process for PDS, enter the following command:

```
EXC:ENVIR:UPROC,FN"/bin/timer",ARGS"120"!
```

4. FILES

/bin/timer
/bin/ps

TOUCH

***** See Warning in Section 1.1 *****

1. NAME

touch - Update access and modification times of a file

2. SYNOPSIS

touch [-amc] [mmddhhmm[yy]] files

3. DESCRIPTION

Touch causes the access and modification times of each argument to be updated. If no time is specified [see *date* (1)], the current time is used. The **-a** and **-m** options cause touch to update only the access or modification times, respectively (default is **-am**). The **-c** option silently prevents *touch* from creating the file, if it did not previously exist.

The return code from *touch* is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

4. SEE ALSO

date(1)

TR

***** See Warning in Section 1.1 *****

1. NAME

tr - Translate characters

2. SYNOPSIS

tr [-cds] [string1 [string2]]

3. DESCRIPTION

Tr copies the standard input to the standard output with substitution or deletion of selected characters. Input characters found in *string1* are mapped into the corresponding characters of *string2.* Any combination of the options **-cds** may be used:

-c	Complements the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 001 through 377 octal.
-d	Deletes all input characters in *string1 .*
-s	Squeezes all strings of repeated output characters that are in *string2* to single characters.

The following abbreviation conventions may be used to introduce ranges of characters or repeated characters into the strings:

[a-z]	Stands for the string of characters whose ASCII codes run from character **a** to character **z,** inclusive.
[a * n]	Stands for *n* repetitions of **a**. If the first digit of *n* is **0**, *n* is considered octal; otherwise, *n* is taken to be decimal. A zero or missing *n* is taken to be huge; this facility is useful for padding *string2 .*

The escape character \ may be used as in the shell to remove special meaning from any character in a string. In addition, \ followed by 1, 2, or 3 octal digits stands for the character whose ASCII code is given by those digits.

The following example creates a list of all the words in *file1* and deposits the list, one per line, in *file2*, where a word is taken to be a maximal string of alphabetics. The strings are quoted to protect the special characters from interpretation by the shell; 012 is the ASCII code for newline.

```
tr -cs "[A-Z][a-z]" "[\012*]" <file1 >file2
```

4. SEE ALSO

ed(1), sh(1).
ascii(5) in the *UTS Programmer Reference Manual*.

5. BUGS

Will not handle ASCII **NUL** in *string1* or *string2 ;* always deletes **NUL** from input.

TRUE

***** See Warning in Section 1.1 *****

1. NAME

true, false - Provide truth values

2. SYNOPSIS

true

false

3. DESCRIPTION

True does nothing successfully. They are typically used in input to *sh* (1) such as the following:

- while true
- do
- command
- done

False does nothing unsuccessfully.

4. SEE ALSO

sh(1)

5. DIAGNOSTICS

True has exit status zero; *false* nonzero.

TTY

***** See Warning in Section 1.1 *****

1. NAME

tty - Get the terminal name

2. SYNOPSIS

tty [-s]

3. DESCRIPTION

Tty prints the pathname for the terminal of the user. The **-s** option inhibits printing of the terminal pathname, allowing one to test only the exit code.

4. EXIT CODES

2	Invalid options were specified
0	Standard input is a terminal
1	Otherwise.

5. DIAGNOSTICS

"not a tty" if the standard input is not a terminal and **-s** is not specified.

UDGNNM

***** See Warning in Section 1.1 *****

1. NAME

udgnnm - releases special diagnostic filename

2. SYNOPSIS

udgnnm unit-name unit-number

3. DESCRIPTION

Udgnnm releases the special file set up by *dgnnm*. It also frees the unit.

4. FILES

/dev/mount
/tmp/ecdxxxxxx
/tmp/xxx

5. SEE ALSO

dgnnm(1)

6. DIAGNOSTICS

Error numbers are returned and may be found in *MOVEerrcod.h*.

7. LIMITATIONS

The interface buffer used by *dgnnm* is passed to this program in */tmp/ectbuf*. If this is lost, there is no way to free the unit.

UMOUNT

***** See Warning in Section 1.1 *****

1. NAME

umount

3. DESCRIPTION

See *mount*.

UNAME

***** See Warning in Section 1.1 *****

1. NAME

uname - Print name of current *UNIX*® system

2. SYNOPSIS

uname [-snrva]

3. DESCRIPTION

Uname prints the current system name of the *UNIX*® system on the standard output file. The main use of *uname* is to determine the system one is using. The following options cause the selected information to be printed:

-s	Print the system name (default).
-n	Print the node name (the node name may be a name that the system is known by to a communications network).
-r	Print the operating system release.
-v	Print the operating system version.
-a	Print all the above information.

Arguments, not recognized, default the command to the **-s** option.

UNCOMPRESS

***** See Warning in Section 1.1 *****

1. NAME

uncompress

3. DESCRIPTION

See *compress*.

URUN

***** See Warning in Section 1.1 *****

1. NAME

urun

3. DESCRIPTION

See *run*.

UUCHECK

1. NAME

uucheck - Check the uucp files, directories, and related information (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

uucheck [*-v*]

3. DESCRIPTION

Uucheck checks for the presence of the uucp system required files and directories. *Uucheck* also does error checking of the *Permissions* file (*/etc/uucp/Permissions*) and the *Config* file (*/etc/uucp/Config*). UUCP related entries in the password file (*/etc/passwd*) are also checked.

Uucheck is usually executed during package installation but can be run any time to verify changes made to the *Permissions* and *Config* files. Note that **uucheck** can only be used by the super user or uucp.

The verbose option (**-v**) prints a detailed explanation of how the UUCP system interprets the *Permissions* and *Config* files.

4. HEADER FILES

none

5. FILES

/etc/passwd	System password file
/etc/uucp/Permissions	Contains information regarding valid user and file access permissions
/etc/uucp/Config	UUCP configuration file.

6. SEE ALSO

uucp(2C), **uustat**(2C), **uux**(2C)

7. DIAGNOSTICS

none

8. LIMITATIONS

none

UUCLEANUP

1. NAME

/usr/lib/uucp/uucleanup, /usr/lib/uucp/uuclean.sh - Cleanup UUCP spooler/related directories (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

/usr/lib/uucp/uucleanup [*options*]
/usr/lib/uucp/uuclean.sh [*options*]

3. DESCRIPTION

3.1 UUCLEANUP

Uucleanup scans the *spool* directories for old files and takes the following appropriate action to remove them in a useful way:

(1) Informs the requester, if local, of send/receive requests for systems that have not contacted the local system.

(2) Returns undeliverable mail to the sender, if local.

(3) Removes all other files.

In addition, there is a provision to warn users of requests that have been waiting for a given number of days (default 1 day).

NOTE: The *uucleanup* command executes **ALL** cleanup strategies regardless of whether the specific option is specified or not.

If a specific option is omitted, *uucleanup* uses the default time for that option.

This program is typically started by the shell *uuclean.sh*, which should be started by **cron**(2G).

The options for *uucleanup* are as follows:

-C#*days*	Removes any *C.files* whose age is greater or equal to #*days* days and send appropriate information to the requester (default 7 days, maximum 180 days).
-D#*days*	Removes any *D.files* whose age is greater or equal to #*days* days (default 7 days, maximum 180 days).
-m*string*	Include the string in the warning message that is generated by the **-W** option.
-o#*days*	Delete ``other'' files whose age is more than #*days* days (default 2 days, maximum 180 days).
-s*system*	Executes only the *spool* directory for the system. Otherwise, all *spool* directories for all systems are cleared.
-W#*days*	If local, any *C.files* whose age is greater than or equal to #*days* days cause a mail message to be sent to the requester, warning of the delay in contacting the remote. The message includes the JOBID. The administrator may include a message line telling to whom to call to check the problem (**-m** option).

NOTE: The time specified for *-C* should be greater than the time for *-W*. Otherwise, no message is generated (default 1 day, maximum 180 days).

-X*#days*	Any *X.files* whose age is greater or equal to *#days* days are removed. (Any related *D.files* are taken care of by *D.file* processing (default 2 days, maximum 180 days).

NOTE: For the previously listed options, *#days* must be greater than 0.

3.2 UUCLEAN.SH

Uuclean.sh is a shell script that performs various cleanup activities on files and directories related to the **uucp** programs. **uuclean.sh** is usually run by the **cron** process. The */usr/lib/crontab* entry usually looks like the following:

```
15 4 * * * /bin/su uucp -c "/usr/lib/uucp/uuclean.sh 2"
```

Uuclean.sh does the following clean up:

(1) Saves and then removes files in */etc/bwm/uucp/.Admin* (except SQFILE) in the 5E14 Software Release or */var/uucp/.Admin* (except SQFILE) in the 5E15 and Later Software Releases

(2) Saves and then removes log and status files in the following files:

5E14 Software Release	**5E15 and Later Software Release**
/etc/bwm/uucp/.Log	*/var/uucp/.Log*
or	or
/etc/bwm/uucp/.Status	*/var/uucp/.Status*

(3) Calls **uucleanup** to clean up *.X*, *.D*, and *.C* files in */etc/bwm/spool/uucp* in the 5E14 Software Release or */var/spool/uucp* in the 5E15 and Later Software Releases. It also removes any miscellaneous, empty directories.

(4) Removes any old files from the **uucp** working directories

(5) Removes any old files from */etc/bwm/spool/uucppublic* in the 5E14 Software Release or */var/spool/uucppublic* in the 5E15 and Later Software Releases.

(6) Removes any old save log and status files from */etc/bwm/uucp/.Old* in the 5E14 Software Release or */var/uucp/.Old* in the 5E15 and Later Software Releases.

(7) Removes all old mail messages for all logins.

NOTE: **uuclean.sh** executes **ALL** cleanup strategies regardless of whether the specific options are specified or not. If a specific option is omitted, **uuclean.sh** uses the default time for that option.

The options for **uucleanup** are as follows:

-C*#days*	Passed to **uucleanup** (default 7 days, maximum 180 days).
-D*#days*	Passed to **uucleanup** (default 7 days, maximum 180 days).
-G*#days*	Removes any files in */etc/bwm/uucp/.Corrupt* in 5E14 Software Release (*/var/uucp/Corrupt* for 5E15 and Later Software Releases) whose age is greater or equal to *#days* days (default 10 days, maximum 180 days).
-K*#days*	Removes any files in */etc/bwm/uucp/.Workspace* in 5E14 Software Release (*/var/uucp/Workspace* for 5E15 and Later Software Releases) whose age is greater or

	equal to *#days* days (default 1 day, maximum 180 days).
-L*#days*	Removes any files in */etc/bwm/uucp/.Old* for 5E14 Software Release (*/var/uucp/.Old* for 5E15 and Later Software Releases) whose age is greater or equal to *#days* days (default 3 days, maximum 180 days).
-M*#days*	Remove any mail messages from files in */etc/bwm/mail/** in 5E14 Software Release (*/var/mail/** in 5E15 and Later Software Releases) whose age is greater or equal to *#days* days (default 3 days, maximum 180 days).
-N*number_of_logfiles*	Each time **uuclean.sh** is called, it saves log, status, and admin data in a unique logfile in the */etc/bwm/uucp/.Old* directory in the 5E14 Software Release (*/var/uucp/.Old* in the 5E15 and Later Software Releases). The **-N** argument indicates how many of these log files to accumulate before overwriting the older ones (default 100 log files, maximum 180 log files).
-o*#days*	Passed to **uucleanup**. (default 2 days, maximum 180 days).
-P*#days*	Removes any files in */etc/bwm/spool/uucppublic* in the 5E14 Software Release (*/var/spool/uucppublic* in the 5E15 and Later Software Releases) whose age is greater or equal to *#days* days (default 30 days, maximum 180 days).
-Q*#days*	Removes any files in */etc/bwm/uucp/.Sequence*, in the 5E14 Software Release, (*/var/uucp/.Sequence* in the 5E15 and Later Software Releases) whose age is greater or equal to *#days* days (default 30 days, maximum 180 days).
-U*#days*	Removes any files in */etc/bwm/uucp/.Xqtdir* , in the 5E14 Software Release, (*/var/uucp/.Xqtdir* in the 5E15 and Later Software Releases) whose age is greater or equal to *#days* days (default 1 day, maximum 180 days).
-W*#days*	Passed to **uucleanup** (default 1 day, maximum 180 days).
-X*#days*	Passed to **uucleanup** (default 2 days, maximum 180 days).

NOTE: For the previously listed options, #days must be greater than 0.

4. HEADER FILES

none

5. FILES

Software Release 5E14

/etc/bwm/spool/uucp	The *spool* directory
/etc/bwm/uucp	All other uucp directories
/etc/bwm/uucp/.Old	Repository for uuclean.sh logfiles
/usr/mail	The primary mail directory

Software Releases 5E15 and Later

/var/spool/uucp	The *spool* directory
/var/uucp	All other uucp directories

/var/uucp/.Old	Repository for uuclean.sh logfiles
/var/mail	The primary mail directory.
/usr/mail	The secondary mail directory, */var/mail*, does not exist.

6. SEE ALSO

cron(2G), **mail**(2G), **uucp**(2C), **uulog**(2C), **uuname**(2C), **uustat**(2C), **uux**(2C),

7. DIAGNOSTICS

For more information on error messages, refer to document 254-303-106, *System Maintenance Manual*.

8. LIMITATIONS

RTR version of **uucleanup** does not send warning or other cleanup status information to users on remote systems.

UUCP

1. NAME

uucp - ***UNIX*** system to ***UNIX*** system file copy (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

uucp [*options*] source-files destination-file

3. DESCRIPTION

Uucp queues a transfer job for the **uucico** daemon. At a later time, **uucico** executes and copies files named by the *source-file* arguments to the *destination-file* argument. A filename may be a pathname on the local system or may have the following form:

system-name!path-name
(*Where* ***system-name*** *is taken from the list of system names which* **uucp** *recognizes.*)

The *system-name* may also be a list of names such as the following:

system-name!system-name!...!system-name!path-name

In which case, an attempt is made to send the file, via the specified route, only to a destination in PUBDIR. See PUBDIR, in section ``FILES'', for more information. Care should be taken to ensure that intermediate nodes in the route are configured to forward information.

Uucp does not directly cause **uucico** to be executed. The **uucico** command is executed when the remote system accesses the RTR system.

The shell metacharacters, **?**, ***** and **[reg-exp]** appearing in *pathname* are expanded on the appropriate system.

Pathnames can be one of the following:

(1) A full pathname.

(2) A pathname preceded by **\~***user* (where *user* is a login name on the specified system and is replaced by the login directory of the user).

(3) A pathname preceded by **\~/***user* where *user* is a login name on the specified system and is replaced by the login name of the user under PUBDIR (see section ``FILES'').

(4) **\~***/file_name* same as **\~***user/file_name*

(5) Anything else is prefixed by the current directory.

If the result is an erroneous pathname for the remote system, the copy fails. If the *destination-file* is a directory, the last part of the *source-file* name is used.

Uucp preserves the execute permissions across the transmission and gives 0666 read and write permissions [see **chmod**(2)].

The meanings of the available options are as follows:

-C	Make a copy of the source files in the *spool* directory before copying them to the destination.

NOTE: **-C** and **-c** are mutually exclusive. **uucp** aborts if both are specified on the command line.

-c	Do **NOT** make copies of the source files in the *spool* directory before copying them to the destination. Copy directly from the source file. This is the default behavior. ***NOTE:*** **-C** and **-c** are mutually exclusive. **uucp** aborts if both are specified on the command line.
-d	Make all necessary intermediate destination directories for the file copy. This is the default behavior.
-f	Do **NOT** make intermediate destination directories for the file copy.
-g*grade*	Specify a priority *grade* for the transfer; *grade* can be either a single character letter or a single character digit. The ASCII value of the character determines the relative priority. That is, ``a'' has a higher priority than ``b''; ``b'' has a higher priority than ``c'', and so forth. The default is ``N''.
-j	Print the **uucp** job number.
-m	Notify requester that the file was sent.
-n*user*	Notify the *user* on the remote system that the file was sent. This notification is sent via **rmail** on the remote system.
-r	Do not start the file transfer, just queue the job. (For this RTR release, **-r** is the default and CANNOT be overridden by the user with a uucp command line option. This option is accepted for compatibility.)
-s*file*	Report status of the transfer to the *file*. This option is accepted for compatibility, but it is ignored because it is insecure.

4. HEADER FILES

none

5. FILES

Software Release 5E14

/etc/bwm/spool/uucp	The *spool* directory
/etc/bwm/spool/uucppublic	PUBDIR - The default public directory for receiving and sending files
/etc/uucp/Permissions	Contains information regarding valid user and file access permissions
/etc/uucp/Systems	Contains valid systems information.
*/etc/uucp/**	Other UUCP configuration files
*/usr/lib/uucp/**	Other UUCP program files.

Software Releases 5E15 and Later

/var/spool/uucp	The *spool* directory

/var/spool/uucppublic	PUBDIR □ The default public directory for receiving and sending files
/etc/uucp/Permissions	Contains information regarding valid user and file access permissions
/etc/uucp/Systems	Contains valid systems information
*/etc/uucp/**	Other UUCP configuration files
*/usr/lib/uucp/**	Other UUCP program files.

6. SEE ALSO

rmail(2G), **uucleanup**(2C), **uulog**(2C), **uuname**(2C), **uustat**(2C), **uux**(2C),

7. DIAGNOSTICS

uucp returns success (0) if the transfer job was successfully queued.

NOTE: This does not imply that the job will execute successfully nor that the files will be successfully transferred.

For more information on error messages, refer to document 254-303-106, *System Maintenance Manual*.

8. LIMITATIONS

The default list of files that can be copied from and to is severely restricted.

See the */etc/uucp/Permissions* file.

The RTR version of **uucp** does not initiate the transfer of any file to a remote system. All transfers are initiated when the RTR system is contacted by the remote system.

UULOG

1. NAME

uulog - UUCP log file dump facility (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

uulog [*-U* | *-u* | *-X* | *-x*] [*-nnn*] [*-s system* | *-f system*]

3. DESCRIPTION

Uulog is used to examine the log files for one of the basic networking commands, **uucp**, **uux**, **uucico**, or **uuxqt**. *Uulog* optionally displays all log entries, all log entires for a given system, or just the last **nnn** entires. If just one system is requested, *uulog* can optionally continue to display newly generated log entires of a given network command.

The log files are stored in */etc/bwm/uucp/.Log* for the 5E14 Software Release (*/var/uucp/.Log* for 5E15 and Later Software Releases).

The meanings of the available options are as follows:

-f*system*	Same as *-s system*; except after displaying the requested number of lines, **uulog** continues to monitor the log files and prints each new log file entry as it is entered into the log file.
-n*nn*	Only the last "nnn" lines of each log file is displayed. If **-nnn** is omitted, all lines are displayed, unless the **-f** option is specified.
-s*system*	Only log files for system *system* are displayed. If **-s** *system* is omitted, log files for all systems are displayed.
-u	Displays the requested entries for the **uucico** command. This is the default behavior.
-U	Displays the requested entries for the **uucp** command.
-x	Displays the requested entries for the **uuxqt** command.
-X	Displays the requested entries for the **uux** command.

For details regarding the meaning of the individual log entries, refer to document 254-303-106, *System Maintenance Manual*.

4. HEADER FILES

none

5. FILES

Software Release 5E14

/etc/bwm/spool/uucp	The *spool* directory
/etc/bwm/uucp/.Log/cmd/sys	The log file directory for the specific command is *cmd* and for the system the command is *sys*.
*/usr/lib/uucp/**	Other UUCP program files.

Software Releases 5E15 and Later

/var/spool/uucp	The spool directory
/var/uucp/.Log/cmd/sys	The log file directory for the specific command is *cmd* and for the system the command is *sys*.
*/usr/lib/uucp/**	Other UUCP program files.

6. SEE ALSO

mail(2G), **uucleanup**(2C), **uucp**(2C), **uuname**(2C), **uustat**(2C), **uux**(2C)

7. DIAGNOSTICS

none

8. LIMITATIONS

The **uulog** does not examine log files saved by **uuclean.sh**.

UUNAME

1. NAME

uuname - UUCP node name (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

uuname [*-l* **|** *-m*]

3. DESCRIPTION

Uuname comman lists the names of all the systems known to **uucp**.

If the **-l** option is used, the local system name is printed.

If the **-m** option is used, the UUCP *spool* mount point is printed.

4. HEADER FILES

none

5. FILES

/etc/uucp/Systems	The file containing the system names.

6. SEE ALSO

mail(2G), **uucleanup**(2C), **uucp**(2C), **uuname**(2C), **uustat**(2C), **uux**(2C)

7. DIAGNOSTICS

none

8. LIMITATIONS

none

UUSETUP

1. NAME

/usr/lib/uucp/uusetup.sh - Setup and configure UUCP files (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

/usr/lib/uucp/uusetup

3. DESCRIPTION

Uusetup.sh interactively asks the administrator questions about their UUCP setup and enters appropriate information into the configuration files */etc/uucp/Config*, */etc/uucp/Systems*, and */etc/uucp/Permissions*. *Uusetup.sh* create any of the preceding configuration files, if they do not already exist. If the */etc/uucp/Permissions* file does not exist but the */etc/uucp/.permissions* does exist, which is how the UUCP software is distributed, it copies the *.permissions* file to *Permissions*.

uusetup.sh must be run by *root* as it creates new logins and sets passwords for them.

Uusetup.sh configures the following items:

(1) Creates the UUCP work directories if necessary. Set the ownership and permissions on the directories and command files.

(2) Checks for and creates, if necessary, the files */etc/uucp/Devices*, */etc/uucp/Dialers*, */etc/uucp/Dialcodes*, which must exist but whose contents are ignored.

(3) If a uucp account is not present, *uusetup.sh* creates the account and home directory and forces the user to set its password.

(4) If the */etc/uucp/Config* file is not present, it creates one with default settings, if the user requests. This file must exist for *uusetup.sh* to continue. It sets the MAILFLAG depending on user input.

(5) The script requests the user to enter names of remote systems. The system name is entered into the */etc/uucp/Systems* file. For each system entered, the user has to choose a login name which that system uses for UUCP file transfers. The login name is entered into the */etc/passwd* file, and the user is forced to set the password for the new account. The system name and login name are used to construct an entry which is added to the */etc/uucp/Permissions* file. The user enters an empty system name to indicate no more systems are to be entered.

(6) Creates the file */etc/uucp/Maxuuxqts* containing the recommended value of 1.

(7) Adds a line to the */etc/bwm/mail/MAXSIZE* file, for the 5E14 Release, and the */var/mail/MAXSIZE* file, for the 5E15 and Later Releases, limiting the mail size for the uucp account to 128k bytes, if no such line is already present.

(8) Check for a **cron** entry to run **uuclean** in the *5ESS*® switch and RTR standard places for crontab files.

(9) Run **uucheck.sh**(2C) for a final verification of the configuration.

4. HEADER FILES

none

5. FILES

Software Release 5E14

/etc/passwd	The password file
/etc/uucp/Config	The UUCP master configuration file
/etc/uucp/Systems	The remote systems that may connect to the 3B21D
/etc/uucp/Permissions	The correspondence between systems and login names
/etc/uucp/Maxuuxqts	The number of concurrent uuxqt processes that may be executing (should be 1)
/etc/uucp/Dialers	Must exist but whose contents are ignored
/etc/uucp/Dialcodes	Must exist but whose contents are ignored
/etc/uucp/Devices	File that must exist but whose contents are ignored
/etc/bwm/mail/MAXSIZE	The per-user limits on mail files
/etc/bwm/spool/uucppublic	UUCP public directory
*/etc/bwm/spool/uucp/**	UUCP job queue directories
*/etc/bwm/uucp/**	UUCP work and administrative directories
*/etc/bwm/spool/locks/**	UUCP lock files.

Software Releases 5E15 and Later

/etc/passwd	The password file
/etc/uucp/Config	The UUCP master configuration file
/etc/uucp/Systems	The remote systems that may connect to the 3B21D
/etc/uucp/Permissions	The correspondence between systems and login names
/etc/uucp/Maxuuxqts	The number of concurrent uuxqt processes that may be executing (should be 1)
/etc/uucp/Dialers	Must exist but whose contents are ignored
/etc/uucp/Dialcodes	Must exist but whose contents are ignored
/etc/uucp/Devices	File that must exist but whose contents are ignored
/var/mail/MAXSIZE	The per-user limits on mail files
/var/spool/uucppublic	UUCP public directory
*/var/spool/uucp/**	UUCP job queue directories
*/var/uucp/**	UUCP work and administrative directories
*/var/spool/locks/**	UUCP lock files.

6. SEE ALSO

For more information on error messages, refer to document 254-303-106, *System Maintenance Manual*.

7. DIAGNOSTICS

If successful, **uusetup.sh** returns 0; otherwise, **uusetup.sh** returns non-zero.

8. LIMITATIONS

During its execution, **uusetup.sh** locks the */etc/passwd* file to prevent corruption of the */etc/passwd* file caused by other users executing the **passwd** or another **uusetup.sh** command at the same time.

uusetup.sh expects that the */etc/uucp/Permissions* file is in the format that it creates. Hand editing this file may confuse **uusetup.sh** or defeat some of its security checking.

UUSTAT

1. NAME

uustat - UUCP status inquiry and job control (For Software Releases **5E14**, **5E15** and Later)

2. SYNOPSIS

uustat [*-a*] [*-sSYSTEM*] [*-uUSER*]
uustat [*-q* | *-m* | *-kJOBID* | *-rJOBID* | *-p*]

3. DESCRIPTION

Using input format 1, *uustat* provides information about jobs queued by either **uucp** or **uux**. Using input format 2, *uustat* can be used to get a variety of information to cancel a particular job or to update the time stamps for a particular job.

The meanings of the available options are as follows:

-a	Reports the status of all jobs currently in the queue, not just the job for the current user.
-k*jobid*	Removes the job, *jobid*, from the queue. This cancels the job. The only login IDs that have permission to remove the job are the creator, the uucp, or root.
-m	Print a list of all the systems known to either **uucp** or **uux** and if jobs are queued for any system, provide a summary of those jobs.
-p	Executes a ``**ps -l**'' for all active communication processes associated with **uucp** or **uux**.
-q	Provides a job summary for any system with jobs in the queue.
-r*jobid*	Touches all the files associated with job, *jobid*. This rejuvenates all the time stamps for the job. This can be used to prevent jobs from being removed by **uucleanup**. The user must be the creator of the job or either the uucp or root ID to do this.
-s*sys*	Provides a detailed status for each job queued for system, *sys*.
-u*userid*	Provides a detailed status for each job queued by user, *userid*.

When no options are given, *uustat* outputs the details status of all jobs queued by the current user.

The format of the detailed status report (**-a** option) is as follows:

```
jobid date/time req-type sys-name user-name file-size file-name uucp-job
                                   - or -
jobid date/time req-type sys-name user-name command uucp-job
```

Where:	*jobid*	The job number assigned by **uucp** or **uux**.
	date/time	Either when the job was initiated or when the job time stamps were last updated by **uustat -r jobid**.
	req-type	Either ``S'' for sending or ``R'' for receiving.
	sys-name	The system name of the remote system.
	user-name	The user that initiated the job.
	file-size	The size, in bytes, of the file being transferred.
	file-name	The name of the file being transferred.
	command	The command to be executed on the remote system.

The format of the summary status report (**-q** option) is as follows:

```
sys-name Ccnt(Cage) Xcnt(Xage) last-d/t Status-info
```

Where:	*sys-name*	The system name of the remote system.
	Ccnt	Number of transfer jobs. If zero, this field is blank.
	Cage	Age, in days, of the oldest transfer job. If zero, this field is blank.
	Xcnt	Number of remote execution jobs. If zero, this field is blank.
	Xage	Age, in days, of the oldest remote execution job. If zero, this field is blank.
	last-d/t	The date/time of the last contact with the remote system.
	Status-info	This is a text field which gives the status of the communication with the remote system. Information provided indicates if the system has a lock file, if the communication is currently active, if the communication has failed and why and how long until the next attempt to retry the transmission.

For a more detailed description of the Status-info field, refer to document 254-303-106, *System Maintenance Manual*.

4. HEADER FILES

none

5. FILES

Software Release 5E14

/etc/bwm/spool/uucp	The *spool* directory
/etc/bwm/spool/uucppublic	PUBDIR - The public directory for receiving and sending files
/etc/uucp/Permissions	Contains information regarding valid user and file access permissions
/etc/uucp/Systems	Contains valid systems information
*/usr/lib/uucp/**	Other UUCP program files.

Software Releases 5E15 and Later

/var/spool/uucp	The *spool* directory
/var/spool/uucppublic	PUBDIR - The public directory for receiving and sending files
/etc/uucp/Permissions	Contains information regarding a valid user and file access permissions
/etc/uucp/Systems	Contains valid systems information
*/usr/lib/uucp/**	Other UUCP program files.

6. SEE ALSO

mail(2G), **uucleanup**(2C), **uucp**(2C), **uulog**(2C), **uuname**(2C), **uux**(2C)

7. DIAGNOSTICS

For error information, refer to document 254-303-106, *System Maintenance Manual*.

8. LIMITATIONS

none

UUX

1. NAME

uux - *UNIX* system to *UNIX* system command execution (For Software Releases **5E14**, **5E15**and Later)

2. SYNOPSIS

uux [*options*] command-string

3. DESCRIPTION

Uux queues a remote execution job for the **uuxqt** daemon. At a later time, **uuxqt** executes and gathers zero or more files from various systems, executes the specified command on the specific system and then, if requested, sends the resultant standard output to a file on a specified system.

NOTE: For obvious security reasons, most systems limit the list of commands that can be executed as a result of an incoming request generated by *uux* . The list of permitted commands is controlled by the */etc/uucp/Permissions* file.

The *command-string* is made up of one or more arguments that look like a shell command line, except that the initial command and any of the filenames may be prefaced by *system-name.*. A null *system-name* is interpreted as the local system.

Pathnames can be one of the following:

(1) A full pathname.

(2) A pathname preceded by **\~***user*. Where *user* is a login name on the specified system and is replaced by the login directory of the user.

(3) **\~***/file_name* same as **\~***user/file_name*

(4) Anything else is prefixed by the current directory.

Uux does not directly cause **uuxqt** to be executed. **uuxqt** is executed when the remote system accesses the RTR system.

Any special shell characters such as the less than symbol (**<**), the greater than symbol (**>**), the semicolon (**;**), or the pipe **|** should either be escaped or quoted either by quoting the entire *command-string* or quoting the special characters as individual arguments.

Uux instructs **uucico** to have all the required files sent to the execution system. For files that are output files, the filename must be escaped using parentheses. (See Example 2.)

uux instructs **uuxqt** , running on the execution system, to notify the requester if the *command-string* cannot be executed successfully. This response comes via remote mail from the remote system. The requester can also request notification on success or failure of the *command-string*, as indicated in the options that follow.

The meaning of the available options are as follows:

-	Use the standard input as input to the *command-string*. (See **-p**).
-a*name*	Notify *name* upon completion.
-b	Include the standard input in the execution status report when the exit status of

command-string indicates an error.

-C	Make a copy of any required file(s) in the *spool* directory before copying them to the destination. ***NOTE:*** **-C** and **-c** are mutually exclusive. **uux** aborts if both are specified on the command line.
-c	Do **NOT** make any copies the required file(s) in the *spool* directory before copying them to the destination. Copy directly from the required file. (This is the default behavior.) ***NOTE:*** **-C** and **-c** are mutually exclusive. **uux** aborts if both are specified on the command line.
-g*grade*	Specify a priority *grade* for the transfer. *grade* can be either a single character letter or a single character digit. The ASCII value of the character determines the relative priority. That is, 'a' has a lower priority than 'b'; "b" has a lower priority than 'c', and so forth.
-j	Print the **uux** job number.
-n	Do **NOT** notify requester that the execution was completed.
-p	Use the standard input as input to the *command-string*.
-r	Do not start the file transfer, just queue the job. (For this RTR release, **-r** is the default and CANNOT be overridden by the user. This option is being provided for comparability.)
-s*file*	Report status of the transfer to file. This option is accepted for compatibility, but it is ignored because it is insecure.
-z	Send notification to user if the exit status indicates success.

3.1 Example 1:

The following **uux** command causes /usr/phil/motd from sys1 and */etc/motd* from sys2 to be sent to a *spool* work area on sys3 the files are "diff'ed". The output of that diff execution is returned to a file, motd.diff, in the current directory of this system.

```
uux "sys3!diff sys1!/usr/phil/motd sys2!/etc/motd >!motd.diff"
```

3.2 Example 2:

This **uux** command does the same as in Example 1, except this version uses the escaped output file notation.

```
uux sys3!diff sys1!/usr/phil/motd sys2!/etc/motd  (!motd.diff)
```

4. HEADER FILES

none

5. FILES

Software Release 5E14

/etc/bwm/spool/uucp	The *spool* directory
/etc/bwm/spool/uucppublic	PUBDIR - The public directory for receiving and sending files

/etc/uucp/Permissions	Contains information regarding valid user and file access permissions
/etc/uucp/Systems	Contains valid system information
*/usr/lib/uucp/**	Other data and program files.

Software Releases 5E15 and Later

/var/spool/uucp	The *spool* directory
/var/spool/uucppublic	PUBDIR - The public directory for receiving and sending files
/etc/uucp/Permissions	Contains information regarding valid user and file access permissions
/etc/uucp/Systems	Contains valid system information
*/usr/lib/uucp/**	Other data and program files.

6. SEE ALSO

mail(2G), **uucleanup**(2C), **uucp**(2C), **uulog**(2C), **uuname**(2C), **uustat**(2C)

7. DIAGNOSTICS

The **uux** command returns success (0) if the execution job was successfully queued. This does not imply that the job necessarily executes successfully.

For error messages, see document 254-303-106, *System Maintenance Manual*.

8. LIMITATIONS

The list of commands that can be executed should usually be severely restricted.
See the */etc/uucp/Permissions* file.

As previously mentioned, only the first command in the *command-string* may have a *system-name!*. All other commands are executed on the system of the first command. The use of the star (*****) shell metacharacter probably does not do what is expected. The shell tokens **<<** and **>>** are not implemented.

The RTR version of **uux** does not initiate the remote-command execution on a remote system. All remote-command execution is only initiated when the RTR system is contacted by the remote system.

VCP

***** See Warning in Section 1.1 *****

1. NAME

vcp - Volume copy

2. SYNOPSIS

/etc/vcp from to [startingblk] nblocks -

3. DESCRIPTION

Vcp is used to make a contiguous copy of a file. First use falloc(1) to allocate the file, then use vcp to copy a file into the contiguous space. The *from* parameter is an existing file to be the source of the data. The *to* parameter is the previously allocated contiguous file to receive the data. *Nblocks* is the number of blocks of data to copy. If a dash (-) is given, then a dot (.) is printed on the terminal every time 256 blocks are copied.

The *startingblk* option is not normally useful.

VCOMPRESS

***** See Warning in Section 1.1 *****

1. NAME

vcompress - Produce a compressed representation of a target file.

2. SYNOPSIS

vcompress -t target **-d** delta [**-s** source] [**-l** log] [**-c**]

3. DESCRIPTION

Vcompress is used in conjunction with **vexpand (1)** to represent a file (the *target*) in a compressed format that can be reproduced to the original form of the file. *Vcompress* produces a compressed representation (the *delta* file) of the *target* file in one of two ways. First, the *delta* file may be created using a strict compression algorithm, similar to **compress (1)**. In this mode, *compression only,* the target file is simply compressed and no source file is specified. The second mode is *data differencing*; whereby, the *target* file is compared to a specified *source* file, no differences.

Vexpand (1) reproduces the *target* file from the *delta* file, and optionally, the *source* file, if the method of compression by *vcompress* was data differencing.

The command line options are as follows:

□**t** *target*	Specifies the name of the file to be represented in compressed format.
□**d** *delta*	Specifies the name of the file to be generated, which contains the compressed format of the target.
□**s** *source*	Optionally, specifies the name of a source file to be used for data differencing.
□**l** *log*	Optionally, specifies the name of a file where output normally intended for *stdout* and *stderr* is to be redirected.
□**c**	Optionally, specifies to perform special coff file header processing. This processing writes the entire header portion of a coff formatted file to the *delta* file. By doing this, **vexpand (1)** reproduces the header portion of the file by copying it from the *delta* file, instead of using data differencing and the *source* file. This is necessary in the software update, SU, environment where the headers in the SU viewpath are not necessarily byte-for-byte comparable with the headers on a *5ESS*® switch. Specifying this option for files that are not in coff format or for files being compressed using the *compression only* mode has no effect.

File *new1* is a modified version of *previous1*. To compress *new1* into file *delta* by the data differencing method using *previous1* as a base for comparison, execute the following:

```
vcompress □t new1 □s previous1 □d delta
```

To reproduce *new1* from the *delta* file execute the following:

```
vexpand □t new1 □s previous1 □d delta
```

Exhibit 1

Newly created file *newcmd.sh* is to be compressed into a file *delta* without data differencing (that is, there is no existing file that can be used as a base for comparison). Execute the following:

```
vcompress □t newcmd.sh □d delta
```

To reproduce *newchmd.sh* from the *delta* file execute the following:

```
vexpand □t newcmd.sh □d delta
```

Exhibit 2

4. HEADER FILES

sfio.h, vdelta.h.

5. SEE ALSO

vexpand(1), diff(1), compress(1).

6. DIAGNOSTICS

vcompress returns 0 upon successful completion and 1 upon failure.

Upon successful completion, **vcompress** outputs the checksum for the *target* file and optional *source* file to *stdout* (or optionally the file specified by the □l option) in the following format:

```
Target Sum=ttttt Source Sum= sssss
```

*Where **ttttt** and **sssss** are the complete checksums of the corresponding file (equivalent to **sum □r**).*

Upon failure **vcompress** outputs error messages to *stderr* (or optionally the file specified by the □l option). Special attention should be paid to the following message:

```
vcompress:internal error <errnum>
```

An internal error occurred within **vcompress**. The *errnum*, along with the *target* file and any optional *source* file, should be reported to your support organization.

7. LIBRARIES

libvdelta.a, sfio.a.

VEXPAND

***** See Waring in Section 1.1 *****

1. NAME

Vexpand - Produces a compressed representation (copy) of the target file.

2. SYNOPSIS

vexpand [-**t**] target [-**d**] delta [-**s**source][-**l**log]

3. DESCRIPTION

Vexpand is used in conjunction with the **vcompress** command to represent a file (the *target* in a compressed format that can be reproduced to the original form. The **vcompress** command produces a compressed representation (the *delta* file) of the *target* file in one or two ways. First, the *delta* file may be created using a strict compression algorithm, similar to *compress (1)*. In this method, the target file is simply compressed and no source file is specified. The second method is *data differencing*; whereby, the *target* file is compared to a specified *source* file, no differences.

The *vexpand* command reproduces the *target* file from the *delta* file. The optional *source* file is used as a base for target creation and is specified only if the *vcompress* data differencing method was used when creating the *delta* file.

The command line options are as follows:

□**t** *target*	Specifies the name of the file to be created from the compressed format contained in the *delta* file.
□**d** *delta*	Specifies the name of the file containing the compressed format of the target.
□**s** *source*	Optionally specifies the name of a source file to be used as a base for target creation. This option is required if *data differencing* was used when creating the *delta* file.
□**l** *log*	Optionally specifies the name of a file where output normally intended for *stdout stderr* is to be redirected.

File *new1* is a modified version of *previous1*. To compress *new1* into file *delta* by the data differencing method using *previous1* as a base for comparison, execute the following:

```
vcompress □t new1 □s previous1 □d delta
```

To reproduce *new1* from the *delta* file execute the following:

```
vexpand  □t new1 □s previous1 □d delta
```

Exhibit 3

Newly created file *newcmd.sh* is to be compressed into a file *delta* without data differencing (that is, there is no existing file that can be used as a base for comparison). Execute the following:

```
vcompress □t newcmd.sh □d delta
```

To reproduce *newchmd.sh* from the *delta* file execute the following:

```
vexpand □t newcmd.sh □d delta
```

Exhibit 4

4. HEADER FILES

sfio.h, vdelta.h.

5. SEE ALSO

vcompress(1), diff(1), compress(1).

6. DIAGNOSTICS

The *vexpand* command returns 0 upon successful completion and 1 upon failure.

Upon successful completion, *vexpand* outputs the checksum for the newly created target file to *stdout* (or optionally the file specified by the □1 option) in the following format:

```
Target Sum= ttttt
```
*Where **ttttt** is the checksum of the complete target file*
*(equivalent to **sum □r**).*

Upon failure, *stderr* (or optionally the file specified by the □1 option). Special attention should be paid to the following messages:

□ vexpand: Bad Delta

A read error occurred while attempting to read the *delta* file header data (for example, magic number, target size, or window size). The delta file may be corrupted.

□ vexpand: Bad magic number

The magic number in the *delta* file header conflicts with that in *vexpand*. The *delta* may be corrupted or was created by an incompatible version of *vcompress.*

□ vexpand: No source file.

The *delta* file was created using a *source* file and data differencing, but no *source* file was specified on the *vexpand* command line.

□ vexpand: Source file size mismatch.

The size of the *source* file used to create the *delta* disagrees with the size of the *source* file specified on the *vexpand* command line.

□ vexpand: internal error <errnum>.

An internal error occurred within *vexpand.* The *errnum,* along with the *delta* file, and any optional *source* file should be reported to your support organization.

7. LIMITATIONS

None

8. LIBRARIES

libvdelta.a, sfio.a.

VI

***** See Warning in Section 1.1 *****

1. NAME

vi - Screen-oriented (visual) display editor based on ex

2. SYNOPSIS

vi [-**r** file] p-L] [-**w**n] [-**R**] [+command] [-**c** command] file ...

3. DESCRIPTION

The *vi* (visual) editor is a display-oriented text editor based on an underlying line editor *ex (1).* It is possible to use the command mode of *ex* from within *vi*.

When using *vi* , changes made to the file are reflected in what is seen on the terminal screen. The position of the cursor, on the screen, indicates the position within the file.

4. INVOCATION

The following invocation options are interpreted by *vi* :

-r *file*	Recovers *file* after an editor or system crash. If *file* is not specified, a list of all saved files is printed.
-L	Lists the name of all files saved as the result of an editor or system crash.
-w *n*	Sets the default window size to *n.* This is useful when using the editor over a slow speed line.
-R	Read only mode; the **readonly** flag is set, preventing accidental overwriting of the file.
[-c|+] *command*	The specified *ex* command is interpreted before editing begins.

The *file* argument indicates files to be edited.

If a user invokes *vi* on a file, edits the file, and then discovers the file is read-only when trying to save the changes, changes made to the file can be saved by entering the following command:

```
:!chmod u+rw filename <CR>
```

After the command executes, press the RETURN key to continue editing the file; save the changes made to the buffer by entering:

```
:w! <CR>
```

Make more changes to the file or quit the editing session; the changes will be saved.

5. VI MODES

Command	Normal and initial mode. Other modes return to command mode upon completion. ESC (escape) is used to cancel a partial command.
Input	Entered by the following options **a i A I o O c C s S R**. Arbitrary text may then be entered. Input mode is normally terminated with the ESC character or abnormally with interrupt.
Last Line	Reading input for **: / ?** or **!**; terminate with CR to execute, interrupt to cancel.

6. COMMAND MODES

6.1 The following are sample commands:

□□□□	Arrow keys move the cursor
h j k l	Same as arrow keys
i*text*ESC	Insert text *abc*
cw*new*ESC	Change word to *new*
ea*s*ESC	Pluralize word
x	Delete a character
dw	Delete a word
dd	Delete a line
3dd	... 3 lines
u	Undo previous change
ZZ	Exit vi, saving changes
:q!**CR**	Quit, discarding changes
/*text***CR**	Search for *text*
^U ^D	Scroll up or down
:*ex cmd***CR**	Any ex or ed command

6.2 Counts permitted with vi commands

Numbers may be typed as a prefix to some commands. They are interpreted in one of the following ways:

line/column number	z G |
scroll amount	^D ^U
repeat effect	Most of the rest

6.3 Interrupting or canceling a vi command

ESC	End insert or incomplete cmd
DEL	(Delete or rubout) Interrupts
^L	Reprint screen if DEL scrambles it
^R	Reprint screen if ^L is □ key

6.4 File Manipulation

ZZ	If file modified, write and exit; otherwise, exit
:w**CR**	Write back changes
:w!**CR**	Forced write, if permission originally not valid
:q**CR**	quit
:q!**CR**	Quit, discard changes
:e *name***CR**	Edit file *name*
:e!**CR**	Re-edit, discard changes
:e + *name***CR**	Edit, starting at end
:e +*n***CR**	Edit starting at line *n*
:e #**CR**	Edit alternate file
:e! #**CR**	Edit alternate file, discard changes synonym for **:e #**
:w *name***CR**	Write file *name*
:w! *name***CR**	Overwrite file *name*
:sh**CR**	Run shell, then return
:!*cmd***CR**	Run *cmd*, then return
:n**CR**	Edit next file in arglist
:n *args***CR**	Specify new arglist
^G	Show current file and line

6.5 Positioning Within File

^F	Forward screen
^B	Backward screen
^D	Scroll down half screen
^U	Scroll up half screen
*n***G**	Go to the beginning of the specified line (End default) Where *n* is a line number
G	Go to specified line (end default)

Ipat	Next line matching *pat*
?*pat*	Previous line matching *pat*
n	Repeat last / or ?
N	Reverse last / or ?
*IpatI***+** *n*	nth line after *pat*
?*pat***?-** *n*	nth line before *pat*
]]	Next section/function
[[	Previous section/function
(	Beginning of sentence
)	End of sentence
{	Beginning of paragraph
}	End of paragraph
%	Find matching **( ) { or }**

6.6 Adjusting the Screen

^L	Clear and redraw
^R	Retype, eliminate @ lines
zCR	Redraw, current at window top
z-	... At bottom
z.	... At center
*IpatI***z-**	Pat line at bottom
*IpatI***z.**	... At center
z*n***.**	Use *n* line window
^E	Scroll window down 1 line
^Y	Scroll window up 1 line

6.7 Marking and Returning

\`;\`;	Move cursor to previous context
´´	... At first non-white in line
m*x*	Mark current position with letter *x*
\`;*x*	Move cursor to mark *x*
´*x*	... At first non-white in line

6.8 Line Positioning

H	Top line on screen
L	Last line on screen
M	Middle line on screen
+	Next line, at first non-white
-	Previous line, at first non-white
CR	Return, same as +
□**or j**	Next line, same column
□**or k**	Previous line, same column

6.9 Character Positioning

^	First non-white
0	Beginning of line
$	End-of-line
h or □	Backwards
l or □	Forward
^H	Same as □
space	Same as □
f*x*	Find *x* forward
F*x*	**f backward**
t*x*	**upto** *x* **forward**
T*x*	**back upto** *x*
;	Repeat last **f F t** or **T**
,	Repeat inverse of last **f F t** or **T**
*n***|**	Move to column *n*
%	Find matching **({)** or **}**

6.10 Words, Sentences, Paragraphs

w	Word forward
b	Back word
e	End of word
)	To next sentence
}	To next paragraph
(	Back sentence
{	Back paragraph
W	Blank delimited word
B	Back **W**
E	To end of **W**

6.11 Corrections During Insert

^H	Erase last character
^W	Erase last word
erase	Your erase, same as **^H**
kill	Your kill, erase input this line
\	**Quotes ^H**, your erase and kill
ESC	Ends insertion, back to command
DEL	Interrupt, terminates insert
^D	Backtab over *autoindent*
0^D	**Backtab to beginning of line;**
	reset left margin of *autoindent*
^^D	Caret (**^**) followed by control-d (**^D**); backtab to beginning of line; do not reset left margin of *autoindent*
^V	Quote non-printing character
□**^D**	Kill *autoindent*, save for next

6.12 Insert and Replace

a	Append after cursor
i	Insert before cursor
A	Append at end-of-line
I	Insert before first non-blank
o	Open line below
O	Open above
r*x*	Replace single char with *x*
R*text* **ESC**	Replace characters

6.13 Operators

Operators are followed by a cursor motion and affect all text that would have been moved over. For example, since **w** moves over a word, **dw** deletes the word that would be moved over. Double the operator, that is, **dd** to affect whole lines.

d	Delete
c	Change
y	Yank lines to buffer
<	Left shift
>	Right shift
!	Filter through command
=	Indent for -2LISP

6.14 Miscellaneous Operations

C	Change rest of line (**c$)**
D	Delete rest of line (**d$)**
s	Substitute characters (**cl)**
S	Substitute lines (**cc)**
J	Join lines
x	Delete characters (**dl**)
X	... Before cursor (**dh)**
Y	Yank lines (**yy)**

6.15 Yank and Put

"Put" inserts the text most recently deleted or yanked. However, if a buffer is named, the text in that buffer is put instead.

3yy	Yank 3 lines
3yl	Yank 3 characters
p	Put back text after cursor
P	Put before cursor
*"x***p**	Put from buffer *x*
*"x***y**	Yank to buffer *x*
*"x***d**	Delete into buffer *x*

6.16 Undo, Redo, Retrieve

u	Undo last change
U	Restore current line
.	Repeat last change
*"d***p**	retrieve *d*'th last delete.

7. AUTHOR

The *vi* editor was developed by The University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science.

8. FILES

/tmp/Ex?????	Temporary file used by *vi* during an edit session.
/tmp/Ex?????	Recoverable edited file preserved as a result of a killed edit session.
/usr/lib/exstrings	Data file containing *vi* output messages.

9. WARNING

Because the **crypt(1)** command is not supported, the **-x** and **-C** encryption options are not available under RTR.

Only **vt100** family of terminals is supported.

10. CAVEATS

Software tabs using **^T** works only immediately after the *autoindent.*

Left and right shifts on intelligent terminals do not make use of insert and delete character operations in the terminal.

If a file contains only a newline and an attempt is made to write the file, the editor removes the newline and the file contains zero characters.

If the file being edited has no characters (that is, zero length) and is exited using **ZZ**, the file does not write. Exiting using **:wq** does create a zero length file.

WC

***** See Warning in Section 1.1 *****

1. NAME

wc - Word count

2. SYNOPSIS

wc [-lwc] [names]

3. DESCRIPTION

Wc counts lines, words, and characters in the named files or in the standard input if no *names* appear. It also keeps a total count for all named files. A word is a maximal string of characters delimited by spaces, tabs, or new lines.

The options **l, w**, and **c** may be used in any combination to specify that a subset of lines, words, and characters are to be reported. The default is **-lwc**.

Names specified on the command line are printed along with the counts.

WHO

***** See Warning in Section 1.1 *****

1. NAME

who - Who is on the system

2. SYNOPSIS

who [file]

who am i

3. DESCRIPTION

Who can list the name of the user, terminal line, login time, elapsed time since activity occurred on the line, and the process-ID of the command interpreter (shell) for each current *UNIX*® system user. The **who** command examines the **/etc/utmp** file to obtain information. If *file* is given, that file is examined. Usually, *file* is **/etc/wtmp**, which contains a history of all the logins since the file was last created.

Who with the **am i** option identifies the invoking user.

4. FILES

/etc/utmp
/etc/wtmp
/etc/inittab

5. SEE ALSO

date(1), login(1), mesg(1), su(1)

WRITE

***** See Warning in Section 1.1 *****

1. NAME

write - Write to another user

2. SYNOPSIS

write user [line]

3. DESCRIPTION

Write copies lines from a terminal to that of another user. When first called, **write** sends the message to the intended receiver:

```
Message from your login (tty ??) [ date ] ...
```

When *write* has successfully completed the connection, two bells are sent to the terminal of the sender to indicate that the typed message is being sent.

The recipient of the message should write back at this point. Communication continues until an end of file is read from the terminal or an interrupt is sent. At that point, *write* writes **EOT** on the other terminal and exits.

If a user wants to write to another user that is logged in more than once, the *line* argument is used to indicate the line or terminal to which the message is to be sent (for example, **tty00**); otherwise, the first instance of the intended user, found in **/etc/utmp**, is assumed and the following message posted:

user is logged on more than one place.

You are connected to "*terminal*".

Other locations are:

terminal

Permission to write may be denied or granted by use of the *mesg(1)* command. Writing to others is normally allowed by default. Certain commands, in particular *pr*(1) disallows messages in order to prevent interference with their output. However, if the user has super user permissions, messages can be forced onto a write inhibited terminal.

If the character **!** is found at the beginning of a line, *write* calls the shell to execute the rest of the line as a command.

The following protocol is suggested for using *write*: when a user starts a *write* to another user, wait for the receiving user to *write* back before continuing to send the message. While participating in a conversation, each person, when expecting a reply, should end each portion of conversation with a distinctive signal [that is, **(o)** for "over"] so that the other user knows when to reply. The signal **(oo)** (for ``over and out'') is suggested when conversation is to be terminated.

4. FILES

/etc/utmp	To find user
/bin/sh	To execute **!**

5. SEE ALSO

mail(1), mesg(1), pr(1), sh(1), who(1)

6. DIAGNOSTICS

"user not logged in" if the user to which an attempt to *write*is not logged in.

ZCAT

***** See Warning in Section 1.1 *****

1. NAME

zcat

3. DESCRIPTION

See *compress*.

GLOSSARY

This section provides acronyms, terms, and abbreviations used in this manual.

-- GLOSSARY --

ACCED
Access Editor

Administration
Administration is a number of related functions with the objective of ensuring the overall provision of service by the *5ESS*® switch. Administration includes the assignment of lines and trunks to the system, memory management, collection of traffic and plant data, provisions for additions and modifications to the switch, service evaluation, and capabilities to control and manage the *5ESS*® switch. The primary objective of administration is to assure that the *5ESS*® switch delivers a high level of quality service to the subscribing customers. This is accomplished by monitoring and evaluating system performance. Potential problems that could cause service deterioration are identified.

AIU
Auxiliary Interface Unit

AM
Administrative Module

The AM is that part of the *5ESS*® switch which performs the part of call processing, administration, and maintenance which cannot be economically distributed to the switching modules. The AM consists of the processor, disk storage, and tape backup units. The AM processor performs the centralized processing functions, high-speed tape, and controls the flow of data between the other dedicated processors distributed throughout the remaining units. The processor functions are fully duplicated (except for the port switch) in order to assure continued processing capability.

ANSI
American National Standards Institute

AP
Applications Processor

ARS
Automatic Route Selection

ASCII
American Standard Code for Information Interchange

ASM
Administative Services Module

AT
Access Tandem

BOC
Bell Operating Company

BORSCHT
Battery Feed, Overvoltage Protection, Ringing, Etc. Functions

BPS
Bits Per Second

A measure of the speed with which data communications can move over a line. Also abbreviated as bps, B/S, and b/s.

BRI
Basic Rate Interface

BST
Basic Services Terminal

CAR
Computer Access Restriction

CC
Control Console

CE
Control Equipment

CI
Critical Indicator

CIC
Customer Information Center

CCITT
International Telephone and Telegraph Consultative Committee

CIU
Craft Interface Unit

CLEI
Common Language Equipment Identification

CM
Communication Module

CNI
Common Network Interface

CO
Central Office

COER
Central Office Equipment Reports

COT
Central Office Terminal

CPE
Customer Premises Equipment

CRT
Cathode Ray Tube

CTAM
Customer Technical Assistance Management

DA
Discontinued Availability

DAP
Display Administration Process

DAS
Data Auxiliary Set

DFC
Disk File Controller

DFI
Digital Facility Interface

DMA
Direct Memory Access

DN
Directory Number

DP
Dial Pulse

DRM
Distinctive Remote Module

DSU
Digital Service Unit

DTMF
Dual Tone Multifrequency

EADAS
Engineering and Administrative Data Acquisition System

A mechanized system that is part of Lucent supplied OSS and is a near real-time data collection and surveillance system. Data transmitted over the data link interface of the *5ESS*® switch is collected and summarized by a data processor system.

EADAS/NM
EADAS/Network Management

EAI
Emergency Action Interface

ECD
Equipment Configuration Database

EIA
Electronics Industry Association

EMACS
EMACS is a screen editor that can be used to create or to edit files using a display terminal.

EOC
Embedded Operations Channel

ESF
Extended Super Frame

FIT
Fully-Initializing Terminal

GMS
Global Messaging Server

Hz
Hertz/Cycles per Second

IC
Interexchange Carrier

IMR
Initial Modification Request

I/O
Input/Output

IOP
Input/Output Processor

IP
Information Product

IPM
Interruptions Per Minute

ISDN
Integrated Services Digital Network

LDFT
Load Format

LED
Light-Emitting Diode

LTD
Local Test Desk

MCC
Master Control Center

Provides craft interface of a *5ESS*® switch office. Consists of a video display terminal with keyboard, ROP, and a key telephone set. The TLWS is also part of the MCC.

MFOS
Multifunctional Operations System

MFT
Metallic Facility Terminal

MIM
Management Information Messages

MML
Man-Machine Language

MMSU
Modular Metallic Service Unit

MTTY
Maintenance Teletypewriter

MTTYC
Maintenance Teletypewriter Controller

NARTAC
North American Regional Technical Assistance Center

NCLK
Network Clock

NCT
Network Control and Timing

NDL
New Data Link

NIT
Non-Initializing Terminal

NM
Network Management

Network Management is a set of real-time procedures aimed at optimizing network performance when the network is under stress due to adverse conditions. The NM provides and operates control and surveillance features that aid in maintaining the network integrity and security during overloads and failures.

OA&M
Operations, Administration, and Maintenance

OAM&P
Operations, Administration, Maintenance, and Provisioning

ODA
Office Data Administration

The mechanism by which initial translation information may be assembled for a *5ESS*® switch. Information from the *5ESS*® switch is entered by using a video terminal and transferred into an ODA computer, assembled, then sent to the *5ESS*® switch.

ODBE
Office Database Editor

ODD

Office Dependent Data

OOS
Out of Service

ORP
Office Records Printer

OS
Operations Systems

OSC
Operator Service Center

OSPS
Operator Services Position System

OSS
Operations Support System

Service that provides the routine operations needed to maintain and engineer the telephone network.

OTC
Operating Telephone Company

A service organization using telephone equipment to provide communications for its customers.

PBX
Private Branch Exchange

Provides business customers service that allows a group of lines to make intragroup calls on an extension dialed basis plus allowing direct inward dialed calls from regular telephone network.

PC
Personal Computer

PC
Peripheral Controller

PICB
Peripheral Interface Control Bus

PIDB
Peripheral Interface Data Bus

PIN
Personal Identification Number

A number used to identify the customer using the ABC service.

PODS
Plain Old Digital Service

POTS
Plain Old Telephone Service

PPS
Pulses Per Second

PROM
Programmable Read-Only Memory

PSU
Packet Switch Unit

RB
Ringback

RBOC
Regional Bell Operating Companies

ROP
Receive-Only Printer

RT
Remote Terminal

RTR
Real-Time Reliable

SCANS
Software Change Administration and Notification System

SCC
Switching Control Center

SCCS
Switching Control Center System

A centralized system that controls the switching operations of many switches.

SCSI
Small Computer System Interface

SD
Schematic Drawing

SM
Switching Module

Equipment consisting of the module controller unit, time slot interchange unit, local digital service unit, and analog and digital interface units. The SM performs 95 percent of all switching performed in the *5ESS*® switch. When an SM is remotely located, it is referred to as an RSM (Remote Switching Module). When an SM is used to support an RSM, it is referred to as an HSM (host SM).

SPCS
Stored Program Control System

STLWS
Supplementary Trunk and Line Work Station

TELCO
Telephone Company

TLWS
Trunk and Line Work Station

TMT

Transmission Maintenance Terminal

TOD
Time Of Day

TOPAS
Trunk, Operations, Provisioning, and Administrative System

TSI
Time Slot Interchange

TSPS
Traffic Service Position System

A type of Traffic Service System having stored program control that provides for the processing and recording of special calls requiring operator assistance.

TT
Touch-Tone

TTF
Transmission Test Facility

TTY
Teletypewriter

TTYC
Teletypewriter Controller

TU
Trunk Unit

VDT
Video Display Terminal

VF
Voice Frequency

List of Figures

List of Tables