Introduction
to ULTRIX
on timessqr.gc.cuny.edu

November 1992

With an account on **timessqr**, you can access thousands of libraries, participate in hundreds of discussion groups, send and receive electronic mail, transfer files, edit and process text, as well as write programs in a number of programming languages. This document is an introduction to computing with DEC/ULTRIX on **timessqr.gc.cuny.edu**. Topics include: accessing **timessqr**; commands for directory, file, and process management; electronic mail; the GNU Emacs display editor; online help; and other topics of interest to the Graduate School community.

Revision/Update Information: Version 1.1

Copyright ©1992 George McClintock III

This document is released under the terms of the GNU General Public License as published by the Free Software Foundation, Inc., 675 Mass Ave., Cambridge, MA 02139, USA. Accordingly:

❶ Permission is granted to copy and distribute this document for no commercial gain, provided that the copyright notice and this permission notice are preserved on all copies.

❷ Permission is granted to copy and distribute modified versions of this document for no commercial gain, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

❸ This document is still evolving, thus the information in this document is subject to change without notice. The author of this document cannot be held responsible for any errors that may appear in this document.

For more information about the GNU General Public License, see the file ``COPYING,'' located in **timessqr**'s /usr/users/emacs/etc directory, or write to the Free Software Foundation.

Comments and suggestions for improvements may be forwarded to **mcc@timessqr.gc.cuny.edu**.

DISCLAIMER: The software described in this document is provided ``**as is**''. No guarantee is made by the author as to the suitability, reliability, security, usefulness, or performance of this software.

This document is **not authorized** by the Computer Center, and should not be construed as a commitment by the Graduate School and University Center of the City University of New York, which assumes no responsibility for any errors that may appear in this document.

Table 1 presents trademarks of the Digital Equipment Corporation:

Table 1 DEC Trademarks

DEC	DECnet	ULTRIX
VAX	VAXcluster	VMS
DECsystem	DECwindows	VT

Contents

PREFACE v

CHAPTER 1 GETTING STARTED 1–1

1.1 COMPUTER CENTER ACCOUNTS 1–1

1.2 ACCESSING TIMESSQR 1–1

1.3 LOGGING IN 1–1

1.4 CHANGE YOUR PASSWORD 1–2

CHAPTER 2 UNIX SHELL COMMANDS 2–1

2.1 ABOUT COMMANDS 2–2

2.2 DIRECTORIES AND FILES 2–2

2.3 DIRECTORY MANAGEMENT 2–3

2.4 FILE MANAGEMENT 2–4

2.5 PROCESS MANAGEMENT 2–7
- 2.5.1 Pipes 2–7
- 2.5.2 Redirection 2–7
- 2.5.3 Redirection with SPSS-X 2–8
- 2.5.4 Pipes with Redirection 2–8
- 2.5.5 Foreground and Background Jobs 2–8
- 2.5.6 Example: Running SAS in the Background 2–8
- 2.5.7 Monitoring Background Processes 2–8
- 2.5.8 Example: Monitoring SAS 2–8
- 2.5.9 Suspending, Moving and Killing Processes 2–9

2.5.10 Wildcards 2–10

2.6 PROTECTION MODES 2–10

2.6.1 Changing File Protection with chmod 2–11

CHAPTER 3 ONLINE HELP 3–1

3.1 APROPOS 3–1

3.2 MAN 3–1

3.3 LEARN 3–2

CHAPTER 4 ULTRIX MAIL 4–1

4.1 SENDING MAIL 4–1

4.2 READING MAIL 4–2

4.3 REPLYING TO MAIL 4–2

4.4 SAVING AND/OR DELETING MAIL 4–2

4.5 QUITTING MAIL 4–2

4.6 REVIEWING MAIL 4–3

4.7 FORWARDING MAIL 4–4

4.8 MAIL ALIASES 4–4

4.9 ABORTING MAIL MESSAGES 4–5

CHAPTER 5 GNU EMACS 5–1

5.1 INTRODUCTION 5–1
5.1.1 Emacs Conventions 5–1

5.2 GETTING STARTED 5–2
5.2.1 Issuing Emacs Commands 5–2
5.2.2 The .emacs File 5–3

5.3 EDIT COMMANDS 5–4
5.3.1 Search and Replace 5–5
5.3.2 Cut and Paste 5–6

5.4 EMACS MAIL AND RMAIL 5–8
5.4.1 Mail 5–8
5.4.2 RMAIL 5–9
5.4.2.1 Invoking RMAIL • 5–9
5.4.2.2 Listing RMAIL Messages • 5–9
5.4.2.3 Reading RMAIL Messages • 5–9
5.4.2.4 Deleting and ``Undeleting'' RMAIL Messages • 5–9
5.4.2.5 Saving RMAIL Messages • 5–10
5.4.2.6 Forwarding and Replying to Messages • 5–10
5.4.2.7 Saving Copies of Mail • 5–10

5.5 GNUS 5–11

5.6 EMACS HELP 5–17

5.7 EMACS DOCUMENTATION 5–17

CHAPTER 6 COMING ATTRACTIONS 6–1

APPENDIX A PINE-MAIL A–1

APPENDIX B GOPHER B–1

APPENDIX C ACCESSING TIMESSQR FROM THE GRACE BLDG. C–1

FIGURES

5–1 A GNU Emacs Buffer 5–3
5–2 A GNUS Buffer 5–13

TABLES

1 DEC Trademarks ii
2 Keyboard Conventions v
3 Command Key Combinations vi
2–1 Directory Management Commands 2–3
2–2 File Management Commands 2–4
2–3 Process Management Commands 2–10
2–4 chmod Option Examples 2–11
4–1 Mail Shell Commands 4–3
4–2 Mail Commands 4–3
4–3 Mail Escape Commands 4–4
5–1 Emacs Command Conventions 5–1
5–2 Cursor Commands 5–4
5–3 File Commands 5–4
5–4 Buffer Commands 5–5
5–5 Formatting Commands 5–5
5–6 Search Commands 5–5
5–7 Kill and Yank Commands 5–7
5–8 Mail Commands 5–8
5–9 RMAIL Commands 5–10
5–10 GNUS Newsgroup Commands 5–14
5–11 GNUS Subject Commands 5–14
5–12 GNUS Article Commands 5–15
5–13 GNUS Post Commands 5–16
5–14 Help Command Key Sequences 5–17

Preface

timessqr.gc.cuny.edu is the name of a DEC 5810 computer manufactured by Digital Equipment Corporation and owned by the Graduate School. Modeled after Bell Laboratories' UNIX system, **timessqr**'s ULTRIX operating system, related tools, and utilities offer a powerful time-sharing, multi-user system to programmers and non-programmers alike. **timessqr** is connected to the Internet network, which provides mail, file transfer service, and remote login capability to thousands of computer systems located all over the world. **timessqr** supports the GNU Emacs and Vi text editors and statistical programs SPSS-X and SAS, as well as the *ftp* and *kermit* file transfer programs. NETNEWS, a.k.a. UseNet news, is also available on **timessqr** through the Emacs GNUS interface.

timessqr has 64 megabytes of main memory and 400 megabytes of storage for users.

Intended Audience

This document is in production for the benefit of the faculty, students and staff of the Graduate School and University Center of The City University of New York.

Conventions

Typeface conventions in this document appear as follows:

- Commands are shown in lowercase **boldface**.
- Options that modify command behavior are shown in **boldface**.
- Arguments to commands are depicted in *oblique* type, and signify variables supplied by the user, such as directory or file names. For example, *filename* is a generic argument indicating a file of any name.

Table 2 presents keyboard conventions used in this document.

Table 2 Keyboard Conventions

Key	Function
<CTRL>	Press the key marked Control.
DEL	Press the DELETE or BACKSPACE key ⌫.
RET	Press the key on the keyboard marked Return.
SPC	Press the SPACEBAR.
TAB	Press the key on the keyboard marked Tab.

Commands combining two (or more) function keys are enclosed in left and right angle brackets (<>); different keys within the same command are separated by a slash (/). Table 3 presents examples of command key combinations.

Table 3 Command Key Combinations

Key	Function
<CTRL/c>	Hold down the Control key and press c.
<Esc/$>	Hold down the Esc key and press $.

Note: If you cannot delete characters from the command line with the BACKSPACE or DELETE keys, use <CTRL/h>.

Document Production Information

This document was produced with VAX DOCUMENT, an electronic publishing system developed by the Digital Equipment Corporation. VAX DOCUMENT is installed on the Graduate School **VAX** (**cunyvms1.gc.cuny.edu**). Requests for information should be forwarded to **mcc@cunyvms1.gc.cuny.edu**.

Related Documents

Additional information can be found in the following documents:

- *The Little Grey Book: An Ultrix Primer* describes introductory features for the new user.
- *The Big Grey Book: The Next Step with Ultrix* contains sections on ULTRIX utilities *awk*, *sed*, and *tbl*, as well as a comprehensive mail guide and a discussion of C-shell programming.
- *The UNIX Programming Environment*, by Brian W. Kernighan and Rob Pike, is required reading for programmers who work in a UNIX environment.
- *UNIX in a Nutshell* is an excellent, and relatively inexpensive, reference guide.

1 Getting Started

1.1 Computer Center Accounts

In order to access **timessqr**, you must open an account at the Computer Center. Pick up an account application in the User Area, or in Room 308. You will be asked to provide a ‘‘userid’’ and a ‘‘password’’, two words used to access multi-user systems. Students must have their application signed by a faculty member from their program. Bring the completed application and your valid Graduate School i.d. to a consultant in the User Area during Computer Center hours. Allow seven working days for processing. If your chosen userid has already been assigned to another user, you will be given a different userid.

1.2 Accessing timessqr

timessqr can be accessed through **Telnet** from any PC or Macintosh computer in the Computer Center. Refer to the Computer Center document, *Telnet*, which explains the procedure for accessing **timessqr**. Ask a consultant for help if you have trouble getting started.

With a modem, **timessqr** can be accessed from a remote location by telephone. Set your telecommunications software parameters as follows:

❶ 8 data bits

❷ 1 stop bit

❸ No parity

timessqr’s telephone number is (212) 642-2761 for modem speeds of 300, 1200, and 2400 baud are supported. The telephone number for V32bis modems is (212) 642-1950.

1.3 Logging In

When you have accessed **timessqr**, **timessqr** will identify itself and prompt you for your userid, as follows:

```
ULTRIX V4.0 (Rev. 179) (timessqr.gc.cuny.edu)

login: foo RET
Password: password RET
```

After entering your userid (**foo** in our example), and pressing RET, **timessqr** will prompt you for your password. Type your password and press RET. As a security feature, your password will not be displayed as you type. Several lines of text will scroll up your screen. These are bulletins about **timessqr**’s availability, the Computer Center schedule, installed software, and other pertinent information. Eventually, the command prompt will appear.

```
Last login: Mon Nov  2 15:33:25 from cunyvms1
ULTRIX V4.0 (Rev. 179) System #1: Thu Feb 21 13:12:54 EST 1991
UWS V4.0 (Rev. 164)

***************************************************************************
   INSTALLED SOFTWARE

 SAS 6.07 SPSSX3.0 & SPSSX4.0   Kermit  GNU Emacs

For more information type: man inform
BACKUPS: Weekly on Saturdays

 SUMMER HOURS: M-Thu 9:00 AM to 5:45PM
***************************************************************************
***************************************************************************

TERM = (vt100)
```

Now press RET and complete your login procedure.

```
Erase set to Backspace
Kill is Ctrl-U
Fri Feb 21 10:17:39 EST 1992
Your current quota is:

Disc quotas for foo (uid 007):
Filsys current   quota   limit  #warns   files   quota   limit  #warns
    /u    1395*   2000    1000             138     150     160

foo(+!)
```

Attention: UNIX is case sensitive. All **timessqr** userids are assigned in *lowercase*. If you should type your userid and password in uppercase, **timessqr** will not recognize you and will return an error message:

```
ULTRIX V4.0 (Rev. 179) (timessqr.gc.cuny.edu)

login: FOO RET
Login incorrect
```

Press RET and login again.

1.4 Change Your Password

If this is your first login to **timessqr**, you should be asked to change your password. Do the following:

❶ Type **passwd** and press RET.

❷ The machine will prompt you for a new password.

❸ Type your new password and press RET.

❹ The machine will then prompt you for verification.

❺ Type your new password again and press RET.

Your new password and verification will not be displayed as you type. If you have not mistyped, **timessqr** will accept your new password and the command prompt will appear.

```
foo(+!) passwd RET
Changing password for foo
Old password: oldpassword RET
Enter new password:  newpassword RET
Verify: newpassword RET
foo(+!)
```

2

UNIX Shell Commands

UNIX philosophy stipulates that the power of a computer resides more in the relationships between programs than in the programs themselves. A UNIX operating system is both a *kernel*, in the words of Kernighan and Pike, ‘‘a program that controls the resources of a computer and allocates them among its users,’’ and a *shell*, ‘‘the program that interprets your requests to run programs....’’ Thus the shell is a *command interpreter*, finding and executing the code in the kernel for commands entered at the keyboard, and a *programming language*, with which command sequences are written to facilitate various tasks.

timessqr supports the Bourne shell (**sh**), a UNIX standard; the Korn shell (**ksh**), an enhanced Bourne shell; the Berkeley UNIX C shell (**csh**); and the T shell (**tcsh**), an enhanced version of the C shell.

The T shell, **timessqr**’s default shell for users, features an interactive command, filename, and userid completion utility. Commands, filenames, and userids can be abbreviated to the first identifying letters. Press Tab to complete the abbreviation. For more information about the T shell, use the **man** command (short for ‘‘manual’’), followed by the keyword *tcsh.* For example:

```
foo(+!) man tcsh RET
```

The Computer Center Network Operations group has provided information about various programs and features supported by **timessqr**. To access this information, use the **man** command, followed by the keywords *inform* or *support*. For example:

```
foo(+!) man inform RET

     timesq-online                                            INFORM(1)


     NAME
          inform - Information relating to current status of various local
          software

     SYNOPSIS
          inform - INTERNET MAIL, SPSSX, SAS, PRINTERS, EMACS, KERMIT, XMO-
          DEM, TCSH, MISC, RELATED BOOKS

     DESCRIPTION

          This help screen announces the status of local software and pro-
          grams that have been modified and/or installed on this machine.
   .
   .
   .
```

2.1 About Commands

timessqr's ULTRIX operating system is an implementation of the UNIX system, thus **timessqr** responds to UNIX commands. There are many UNIX commands and options, far too many for an introduction. We shall be forced to limit our discussion to those commands most needed by the user who is new to **timessqr** or otherwise unfamiliar with UNIX systems. For more information, refer to the *ULTRIX Reference Pages* using the **apropos** and **man** commands described above.

There are four rules to remember about UNIX commands:

➊ **UNIX is case sensitive.**

➋ **UNIX commands have ‘‘options’’ which determine command behavior.**

➌ **UNIX commands can take from zero to several ‘‘arguments.’’**

➍ **UNIX commands are typed in lowercase.**

UNIX is case sensitive. timessqr will recognize *MYFILE*, *MyFile*, and *myfile* as three different files. To avoid confusion, UNIX filenames tend to be written in lowercase.

UNIX commands have ‘‘options’’ which determine command behavior. Options are specified in both upper- and lowercase and are usually, but not always, preceded by a dash (-). For example, **ls -r** lists files sorted in reverse alphabetical order, while **ls -R** recursively lists all files located in the current working directory and in all subdirectories stemming from the current working directory. Options are powerful, and occasionally dangerous. For example, **rm -r** can be used to remove a directory and everything in it, including subdirectories.

UNIX commands can take from zero to several ‘‘arguments.’’ An argument is the recipient of the action executed by the command. Commands are executed when RET is pressed.

UNIX commands are typed in lowercase. Commands typed in uppercase will *not* be recognized by **timessqr**, which will return an error message.

2.2 Directories and Files

Files are stored in directories. Like other operating systems (including DOS and VMS), UNIX uses a hierarchical directory structure. Directory paths resemble an upturned tree in which one directory is the root and all other directories are branches of the root. The root directory is represented by a forward slash (/). In the following diagram, the subdirectories **/usr**, **/u**, and **/tmp** grow from the root, while **/tmp/programs** is a subdirectory of **/tmp**, **/u/guest** is a subdirectory of **/u**, and **/usr/bin** is a subdirectory of the **/usr** subdirectory, etc.

```
          / root
         /|\
        / | \
       /  |  \
      /   |   \
     /    |    \
    /     |     \
   /usr   /u     /tmp
  /      /      /
 /      /      /
/bin   /guest /programs
       \      \
        \      \
        /foo   /cia
```

Users are assigned a ‘‘home’’ directory when their accounts are created. Your home directory, otherwise known as your ‘‘default login’’ directory, is identified by your three-letter userid. For example, if **foo** were a guest on **timessqr**, **foo**’s directory path from the root to the home directory would be **/u/guest/foo**.

2.3 Directory Management

Table 2–1 presents commands and command options for directory management:

Table 2–1 Directory Management Commands

Command	Options	Action
cd		change directory
ls		list files in a directory
ls	**-l**	list files in long format: by mode, number of links, owner, size in bytes, and time of last modification
ls	**-al**	list all files in long format
mkdir		make directory
pwd		print working directory
rmdir		remove directory

- The **cd** command changes directories from your current working directory to the directory you name as an argument to the command. The **cd** command without an argument returns to your home directory. The **cd** command, when followed by two periods (**cd ..**), moves one level up the directory path.
- The **ls** commands lists the files and subdirectories in your current working directory or in a specified directory. The **ls** command has many options, of which **ls -l** and **ls -al** are the most common. The **a** option causes the **ls** command to list files preceded by a period (.), known as ‘‘dotfiles,’’ that are otherwise invisible to the **ls** command. (Dotfiles are used to customize the computing environment, and are treated below.) The following example shows typical output from the **ls -al** command:

```
foo(+!) ls -al
total 367
drwxr-xr-x 11 foo          512 Feb 22 16:31 .
drwxr-xr-x  6 root         512 Feb  6 14:06 ..
-rwxr-xr-x  1 foo          496 Feb 19 21:56 .cshrc
-rwx------  1 foo         3284 Feb 10 13:15 .emacs
-rw-------  1 foo          754 Feb 23 01:51 .history
-rwxr-xr-x  1 foo          553 Feb 19 21:56 .login
-rwx------  1 foo          562 Feb 17 20:09 .mailrc
-rw-------  1 foo          209 Feb 12 20:12 yourfile
-rwxr-xr-x  1 foo          182 Feb 16 20:05 .profile
-rw-------  1 foo          209 Feb  7 05:05 .signature
drwxr-xr-x  3 foo          512 Feb 18 05:09 News
-rw-------  1 foo       300877 Feb 22 16:30 RMAIL
drwxr-xr-x  2 foo          512 Feb 22 01:21 bin
drwx------  2 foo          512 Feb 19 02:26 humanist
drwxr-xr-x  2 foo          512 Feb 22 01:24 sasuser
drwxr-xr-x  2 foo          512 Feb  9 12:48 spssuser
foo(+!)
```

For each file or subdirectory in **foo**'s home directory, the fields in the above listing show from left to right: the protection mode; the number of links; the owner; the size in bytes; the date and time of the creation or most recent modification; the filename. The ''**d**'' at the beginning of a protection mode indicates that the entry is a directory. Protection modes are treated in the discussion of the **chmod** command.

- The **mkdir** command is used to create subdirectories.
- The **pwd** command prints the path of your working directory on the screen. Note that **pwd** does not take an argument. In the example below, **foo** uses the **pwd** command to learn the path from his home directory to the root.

```
foo(+!) pwd RET
/u/guest/foo
foo(+!)
```

- The **rmdir** command is used to delete subdirectories. Subdirectories must be empty before they can be deleted with the **rmdir** command.

2.4 File Management

Table 2–2 presents commands and command options for file management.

Table 2–2 File Management Commands

Command	Options	Action
cat		concatenate files
cat	**-n**	display every line in a file with a line number
cat	**-v**	display non-printing characters
cp		copy files
cp	**-i**	ask for permission to overwrite a previous version of file before copying
cp	**-r**	copy all files and subdirectories to the specified directory

Table 2–2 (Cont.) File Management Commands

Command	Options	Action
grep		search file for pattern or ``regular expression''
head		display file's first ten lines
head	**-n**	display file's first *n* lines
less		scroll file both forward and backward (press SPC to scroll forward; press **b** to scroll backward)
lpr		print files
lpr	**-P0**	print file on line printer
lpr	**-Partemis**	print file on laser printer: *PostScript only*
more		scroll file (press SPC to advance)
more	**+n**	begin scrolling file at line number *n*
more	**+/pattern**	begin scrolling file two lines before *pattern*
mv		move and/or rename file
mv	**-i**	ask for confirmation before moving and/or renaming file if a previous version is to be overwritten
pg		scroll file screen by screen (press RET to advance)
printfort		print SPSS-X and SAS statistical output files on the line printer
rm		remove file
rm	**-i**	ask for confirmation before removing file
sort		sort lines of one or more files
tail		display file's last ten lines
tail	**+n**	display file's last *n* lines

- The **cat** command is often used to display files on the screen. **cat** takes zero or more arguments and displays the contents of each file in the order specified. Arguments are separated by a blank space.
- The **cp** command copies a file to another location or to another file. In the following example, **foo** copies **myfile** from **foo**'s home directory to the */tmp* directory:

```
foo(+!) cp ~/myfile /tmp RET
```

- The **grep** command invokes the ``Global Regular Expression Printer'', used to search for character combinations (or ``strings'') specified on the command line. **grep** filters out unwanted information and displays the specified ``regular expression'' on the screen. The **grep** utility comprises three ``pattern matching'' tools, **grep**, **egrep**, and **fgrep**. In the following example, **foo** uses **grep** to see if **cia** is listed in a database called ``spiesdb'':

```
foo(+!) grep cia spiesdb RET
Name: cia   Identity: NA    Assignment: NA
```

- The **less** command is used to scroll both forward and backward in a file, and accepts numerous commands entered at the colon (:) prompt. **less** is much faster than **more** because it does not read the entire file before starting the display. For a list of the **less** commands's options, type **h** or **H** at the colon (:) prompt.

- The **lpr** command is used to print files. An option must be used to specify the printer:
  lpr -P0 and **lpr -P2** specify the line printers.
  lpr -Partemis specifies the laser printer. Attention: Only PostScript files may be printed with the **lpr -Partemis** command.)

- The **more** command displays files according to various commands entered at the –More– prompt. With **more** subcommands you can scroll backward and forward, perform word searches, edit with the **Vi** editor, and exit at any time. To scroll line by line, press RET; to scroll screen by screen, press SPC. Type **q** to exit **more**. **more** has many options. For a list of **more**'s options, type **h** or **?** at the –More– prompt.

- The **mv** command renames a file, or ‘‘moves’’ it to a new name. **mv** also moves any number of files to another directory. In the following example, **foo** uses the **mv** command to move **filename** to the */tmp* directory.

```
foo(+!) mv filename /tmp RET
```

- The **mv -f** command forces the move and suppresses error messages if an earlier version of *filename* in the */tmp* directory is overwritten. Be careful with the **mv -f** command.

```
foo(+!) mv -f filename /tmp RET
```

- The **printfort** command is used to print statistical output from SPSS-X and SAS, which have FORTRAN printer control characters in the first column. Specify the line printer (**0** or **2**) on the command line, as follows:

```
foo(+!) printfort filename 0 RET

foo(+!) printfort filename 2 RET

foo(+!) printfort -h RET (displays a brief help message)
```

- The **rm** *filename* command deletes *filename*; the **rm -i** *filename* command asks you for confirmation before removing *filename*; the **rm -r** *filename* command, if *filename* names a directory, removes the directory and everything in it, including subdirectories. Be very careful with the **rm** command. Deleted files cannot be recovered.

- The **sort** command is used to sort lines of data from one or more files. **sort** sorts information both alphabetically and numerically.

2.5 Process Management

Everytime you login to **timessqr**, the ULTRIX system creates a ‘‘process,’’ otherwise known as a ‘‘job,’’ and assigns it to your userid. The shell program is your login process. When you enter a command, you are commanding the shell to run a program. When you press RET, the shell starts a new process and runs the commanded program.

A command that reads input from the keyboard (instead of from a specified file) is said to read from ‘‘standard input’’; a command that prints output on the screen writes to ‘‘standard output.’’ An example of a program that reads from standard input is **mail**; **cat** is a program that writes to standard output.

Standard input and standard output can be redirected with ‘‘pipes’’ and ‘‘redirection operators.’’

2.5.1 Pipes

Standard input and standard output (what you see on your screen) are redirected with pipes. Represented by a vertical bar (|), a pipe filters the standard output from one command into another command, or from one program into another program. In the following example, the standard output of the **ls -l** command, a long directory listing, is piped into the **lpr** print command for printing on the line printer:

```
foo(+!) ls -l | lpr -P0 RET
```

2.5.2 Redirection

Standard input and standard output (what you see on your screen) are also redirected with redirection operators, represented by the right and left angle brackets (< and >). In the following example, **foo** uses the **grep** pattern matching utility to search for the word ‘‘poet’’ in the file *intello.txt* by redirecting standard input from the keyboard to the file:

```
foo(+!) grep poet < intello.txt RET
```

The complement to the < redirection operator is the > operator, which redirects standard output to a file rather than to the screen. In the following example, **foo** saves a directory listing by redirecting the standard output of the **ls** command to a file called *ls.out*:

```
foo(+!) ls -agiFlR > ls.out RET
```

Two redirection operators together (>> and <<) are used to append redirected standard output and standard input to files that exist in the destination directory. In the following example, **foo** changes directories to the */tmp* directory, concatenates two files, and redirects the standard output in order to append the two files to a third file already present in **foo**’s *home* directory:

```
foo(+!) cd /tmp RET
foo(+!) cat hisfile herfile >> ~/myfile RET
```

Warning: Redirection operators are dangerous. If you redirect standard output of a command to a file that already exists in your current working directory, you will overwrite the original file with the new output.

2.5.3 Redirection with SPSS-X

Both the **<** and the **>** redirection operators are used with the statistics program **SPSS-X**, as follows:

```
foo(+!) spssx < inputfile.spx > outputfile RET
```

2.5.4 Pipes with Redirection

Pipes and redirection operators can be combined in command sequences. In the following example, **foo** generates a listing of current **timessqr** users with the **who** command, pipes the standard output into the **sort** command for alphabetical sorting, and redirects the sorted standard output into a file named *unowho*:

```
foo(+!) who | sort > unowho RET
```

2.5.5 Foreground and Background Jobs

When you execute a command, your shell process waits for the command to be executed by the system, accepts input from the keyboard, and outputs data to the screen. This *interactive* command procedure is called a ‘‘foreground’’ process. The **mail** utility is an example of a foreground process.

2.5.6 Example: Running SAS in the Background

Many programs do not require input from the keyboard. For example, when the
sas *filename.sas*
command is issued, **timessqr** processes *filename.sas*; if you submit the job as a foreground process, you will be forced to wait while **timessqr** runs your program. It is more efficient to run *non-interactive* processes in the background. An ampersand (&) entered at the end of the command line tells **timessqr** to run the program as a ‘‘background’’ process:

```
foo(+!) sas myfoo.sas -nodms -print myfoo.lis -log myfoo.log & RET
```

2.5.7 Monitoring Background Processes

Background jobs can be monitored with the **jobs** command.

2.5.8 Example: Monitoring SAS

In the following example, **foo** runs a SAS program in the background, then monitors its progress:

```
foo(+!) sas theirfile.sas & RET
[1] 21064
foo(+!) jobs RET
[1]  + Running              theirfile.sas
```

2.5.9 Suspending, Moving and Killing Processes

Sometimes it is necessary to suspend a foreground process in order to perform another task at the keyboard. Press <CTRL/z> to suspend a foreground process. Execute the **fg** command to resume the most recent suspended process. In the following example, **foo** suspends the Emacs editor with <CTRL/z>, mails a file (using the **<** redirection operator), issues the **jobs** command in order to monitor his statistics program, then returns to the Emacs editor with the **fg** command:

```
foo(+!) emacs intello.txt  RET
   .
   .
   .
 CTRL/z
Stopped
foo(+!) mail cia < memo.txt  RET
foo(+!) jobs  RET
[1]  + Running              theirfile.sas
[2]    Stopped              emacs
foo(+!) fg  RET
emacs intello.txt
   .
   .
   .
```

Note: The fg command brings your most recent suspended process to the foreground. If you have several processes running in the background and wish to bring one to the foreground, use the fg command followed by a percent sign (%) and the process number.

Foreground processes are sent to the background with the **bg** command, as follows:

❶ Suspend the foreground process with <CTRL/z>.

❷ Find the job number with the **jobs** command.

❸ Send the job to the background with the **bg** command, followed by a percent sign (%) and the job number.

In the following example, **foo** sends a statistics program running in the foreground to the background:

```
   .
   .
   .
 CTRL/z
Stopped
foo(+!) jobs  RET
[1]    Stopped              theirfile.sas
foo(+!) bg %1  RET
foo(+!)
```

Background processes are brought to the foreground with the **fg** command, followed by a percent sign (%) and the job number.

Processes are terminated with the **kill** command, followed by a percent sign (%) and the job number. In our next example, **foo** ‘‘kills’’ the statistics program, then executes the **jobs** command:

```
foo(+!) kill %1  RET
foo(+!) jobs  RET
[1]    Terminated           theirfile.sas
foo(+!)
```

Table 2–3 presents commands and command options for process management.

Table 2–3 Process Management Commands

Command	Action
CTRL/z	Suspend process
bg %*n*	Move process to the background
fg %*n*	Move process to the foreground
jobs	Monitor status of processes
kill %*n*	Kill process

2.5.10 Wildcards

‘‘Wildcards’’ can be substituted for any character or string of characters used as arguments to commands. The most common wildcard, represented by an asterisk (*), is used as a single argument, or is combined with characters in character strings. In the following example, **foo** uses the asterisk (*) wildcard to locate files in the */tmp* directory that contain the word ‘‘file’’:

```
foo(+!) ls -l /tmp/*file RET
-rwxr-xr-x  2 cia          512 Feb  9 12:48 myfile
-rwxr-xr-x  2 cia          512 Feb  9 12:48 yourfile
-rwxr-xr-x  2 cia          512 Feb  9 12:48 hisfile
-rwxr-xr-x  2 cia          512 Feb  9 12:48 herfile
```

In our next example, **foo** uses the asterisk (*) wildcard and the **>** redirection operator to concatenate the four */tmp/*file* files into one. The **>** redirection operator is also used to write the new concatenated file to **foo**’s home directory:

```
foo(+!) cat /tmp/*file > ~/theirfile RET
```

Warning: Wildcards are dangerous. If you use the asterisk (*) wildcard with the **rm** command, all files located in the current working directory will be deleted.

The question mark (**?**) wildcard is used to represent a single character; two question mark wildcards (**??**) together represent two characters.

2.6 Protection Modes

Multi-user systems necessitate both global file protection and an open environment, where users can share resources and work together. UNIX distinguishes between three user categories: owner; group; and world. Read, write, and execute access is determined for each user category.

The **ls -l** command takes both directories and filenames as arguments, thus in the following example, **foo** uses the **ls -l** command to determine the protection mode of a file owned by user **cia** and located in the */tmp* directory:

```
foo(+!) ls -l /tmp/starbytes
-rwxr-x--x  2 cia          512 Feb  9 12:48 starbytes
```

-rwxr-x–x indicates the protection mode of the file *starbytes*. The first dash (**-**) indicates that *starbytes* is a file, not a directory. **rwx** shows that the owner has read, write, and execute access to the file. **r-x** indicates that members of the owner's group have read and execute access. The final **–x** shows that all other users have execute access to the file.

Read access is necessary to view, copy, or print a file. Renaming, editing, or changing a file in any way requires write access. Command procedures such as shell scripts and programs require execute access.

If the protection mode refers to a directory, read access allows the directory's contents to be listed with the **ls** command; write access indicates that files may be created, modified, or deleted within the directory; execute access is based on the protection mode of the files in the directory.

2.6.1 Changing File Protection with chmod

The **chmod** command allows users to change the access categories of their directories and files. Users cannot access directories and files they do not own unless the owner has granted permission with the **chmod** command. The syntax of the **chmod** command is as follows:

```
chmod [user group others][+ − =][read write execute] filename
```

Table 2–4 presents various options for the **chmod** command.

Table 2–4 chmod Option Examples

chmod	User Access	Action
chmod	**ugo+rwx**	Grant owner, group, and others read, write, and execute access.
chmod	**go-rwx**	Deny group and others read, write, and execute access.
chmod	**g+rw,o-rw**	Grant group read and write access; deny others read and write access.
chmod	**u=rw,g+x,o-r**	Set read and write access for owner; grant group execute access; deny others read access.

In the following example, **cia** moves to the */tmp* directory with the **cd** command, checks the protection of his file *starbytes* with the **ls -l** command, changes the protection mode of the file so that other users can read it, and reissues the **ls -l** command to verify the change:

```
cia(+!) cd /tmp RET
cia(+!) ls -l starbytes RET
-rwxr-x--x  2 cia         512 Feb  9 12:48 starbytes
cia(+!) chmod o+r starbytes RET
cia(+!) ls -l starbytes RET
-rwxr-xr-x  2 cia         512 Feb 19 12:58 starbytes
```

3 Online Help

timessqr's ULTRIX operating system supports three standard UNIX online help programs: **apropos**; **man**; and **learn**.

3.1 apropos

The **apropos** command is used to print those sections of the *ULTRIX Reference Pages* that contain one or more ''keywords'' in the command argument. For example, the **cp** command is used to copy files; if you do not know the name of the **cp** command, issue the **apropos** command with *copy* as the argument and press RET. **apropos** will list all commands containing the keyword ''copy'' in the command description. The number in parentheses printed after commands in **apropos**'s list refers to the chapter in the *ULTRIX Reference Pages* where information about the commands are located.

```
foo(+!) apropos copy RET
bcopy, bcmp, bzero, ffs (3) - bit and byte string operations
copysign, drem, finite, logb, scalb (3m)   - copysign, remainder, exponent mans
cp (1)                    - copy file data
cpio (1)                  - copy file archives in and out
dd (1)                    - copy and convert data
erase, werase (3cur)      - copy blanks into window
get (1)                   - get a copy of SCCS file
pg (1)                    - file perusal filter for soft-copy terminals
rcp (1c)                  - remote file copy
rpc_dup_handle (3ncs)     - make a copy of an RPC handle (client only)
uucp, uulog, uuname (1c)     - unix to unix copy
foo(+!)
```

3.2 man

The **man** command (short for ''manual'') displays the *ULTRIX Reference Pages* on your screen. For specific information about UNIX commands, type **man**, followed by the name of a command, and press RET. (If you do not know the name of the command you need, use the **apropos** command followed by the appropriate keyword.) For example:

```
foo(+!) man cp RET

                                                                cp(1)

   Name
     cp - copy file data

   Syntax
     cp [ -f ] [ -i ] [ -p ] file1 file2

     cp [ -f ] [ -i ] [ -p ] [ -r ] file... directory

     cp [ -f ] [ -i ] [ -p ] [ -r ] directory... directory

   Description
   .
   .
   .
```

man -k is a synonym for the **apropos** command.

For more information about the **man** command, issue the **man** command with *man* as its argument and press RET. For example:

```
foo(+!) man man RET
                                                                   man(1)

   Name
     man - displays manual pages online

   Syntax
     man -k keyword...
     man -f page_title...
     man [ options ] [ -roff_options ]... [ section ] page_title...
     man [ options ] [ section page_title... ]...

     /usr/bin/man [ option ] [ section ] page_title...
     /usr/bin/man [ options ] [ section page_title... ]...
   .
   .
   .
```

3.3 learn

The **learn** command starts an interactive tutorial about UNIX applications. **learn** begins with a list of subjects and instructions for interacting with the tutorial. Courses supported by **learn** include the UNIX file system, Vi editor, and C programming language, as well as macro packages and the *eqn* preprocessor for typesetting mathematical expressions.

```
foo(+!) learn
These are the available courses -
  files
  editor
  vi
  morefiles
  macros
  eqn
  C
If you want more information about the courses,
or if you have never used 'learn' before,
press RETURN; otherwise type the name of
the course you want, followed by RETURN.         RET

files     - basic file handling commands
editor    - text editor; must know about files first
vi        - screen-oriented text editor; must know files first
morefiles - more on file manipulations and other useful stuff
macros    - "-ms" macros for BTL memos & papers;  must know editor
eqn       - typing mathematics;  must know editor
C         - writing programs in the C language;  must know editor

This is probably the proper order, but after you
have the "files" course and know the basics of "editor",
try anything you like.

You can always leave learn band it will go on to the next lesson.

Which subject?
```

4 ULTRIX Mail

Electronic mail allows users to communicate with one another and with computer systems located all over the world. **timessqr** supports a number of Mail User Agents (MUA), including Ultrix Mail, GNU Emacs Mail and RMAIL, as well as Pine-Mail.

This chapter presents basic mail commands for newcomers to **timessqr** using Ultrix Mail. For more advanced mail management technique using this MUA, please see *The Little Grey Book: An Ultrix Primer* and *The Big Grey Book: The Next Step with Ultrix.*

The GNU Emacs Mail and RMAIL programs are discussed in the chapter devoted to the GNU Emacs display editor; Pine-Mail is treated in an appendix.

The ULTRIX MUA is invoked with the **mail** command.

4.1 Sending Mail

To send mail to one or more users:

❶ Type **mail**, followed by the userid and address of each recipient. Leave a blank space between each address. If the recipient is a user on **timessqr**, type only the userid.

❷ After typing **mail** and the userids of those whom you wish to contact, **timessqr** will prompt you for the subject of your message.

❸ Type the subject text, then press RET. You will be prompted for your message.

❹ Type your message, pressing RET at the end of each line of text. (ULTRIX mail will not automatically wordwrap text.)

❺ When you are finished, press RET, type a period (.) on the blank line and press RET again, or press <CTRL/d>. You will be prompted by the **Cc:** prompt.

❻ If you wish to send a carbon copy of your message to additional users, type their userids now, and press RET. If you do not wish to send carbon copies, press RET at the **Cc:** prompt. When you press RET, the mail will be sent.

In the example below, **foo** sends a message to two users, the first located on **timessqr**, the second on another system:

```
foo(+!) mail cia boris@badinoff.kremlin.edu RET
Subject: Intelligence Policy RET
Enter message. Use <ctrl/D> or a period alone on a line to terminate the letter.
Comrads:
Given the new openness sweeping the motherland, classified documents
are now unclassified.  In the interest of national security, let us
reclassify classified documents as TIPTOP SECRET. Tempus fugit.
. RET
Cc: bullwinkle@frostbite.falls.edu RET
foo(+!)
```

4.2 Reading Mail

If there is unread mail waiting for you, **timessqr** will tell you when you login. To read your mail:

❶ Type **mail** with no arguments. **timessqr** will display a list of headers reporting the sender's userid, the time, and the subject of each message, followed by an ampersand (**&**). The ampersand is the mail prompt.

```
"/usr/spool/mail/foo": 3 messages 3 unread
>U  1 you@estude.com Sun Feb 23 11:07  47/1938 "how to do these things..."
 U  2 bullwinkle@frostbite.falls.edu Sun Feb 23 11:23  39/1641 "UNIX Follies"
 U  3 boris@badinoff.kremlin.edu Sun Feb 23 11:25 47/1907 "Re: UNIX Follies"
&
```

❷ To read your messages, press RET. The first message will be displayed on your screen.

❸ If the message is long, the –More– prompt will appear in the lower left corner of your screen.

❹ Press SPC to scroll the message screen by screen.

❺ At the end of the message, the mail prompt (**&**) will reappear. Press RET to read the next message.

4.3 Replying to Mail

Type an uppercase **R** at the mail prompt (**&**) to reply to the message you have just finished reading. The **R** command automatically includes in your reply the address and subject header of the original message. Be careful with the **R** command. If you should enter the command with a lowercase **r**, your reply will be sent not only to the author of the original message, but also to everyone who received a copy of that message.

4.4 Saving and/or Deleting Mail

Type **s** *filename* at the mail prompt (**&**) to save a message in a file.

Type **d** at the mail prompt (**&**) to delete a message. If you have several messages and wish to delete some of them, type the **d** command, followed by the numbers of the messages to be deleted. For example:

```
& d 2 4 6-9 RET
```

If by accident you have deleted a message you need but have not yet quit the mail program, you can undelete the message with the **u** command. For example:

```
& u 2 RET
```

4.5 Quitting Mail

Type **q** at the mail prompt (**&**) to quit the mail program. The **q** command exits mail and saves all undeleted read messages in a file named *mbox*, where they can be reviewed at a later date. Deleted mail is discarded. Unread messages are retained in the **timessqr**'s mailbox.

4.6 Reviewing Mail

To review your mail:

❶ Type the **mail** command with the **-f ~/mbox** option. **timessqr** will display the contents of your *mbox* and the mail prompt (**&**) will appear.

❷ Type **h**. The **h** command lists the messages in your *mbox* by header.

```
foo(+!) mail -f ~/mbox RET
"/u/guest/foo/mbox": 3 messages"
    1 you@estude.com Sun Feb 23 11:07  47/1938 "how to do these things..."
    2 boris@badinoff.kremlin.edu Sun Feb 23 11:23  39/1641 "UNIX Follies"
    3 bullwinkle@frostbite.falls.edu Sun Feb 23 11:25  47/1907 "Re: UNIX Follies"
&
```

❸ To review a message from the header list, type the message number at the mail prompt (**&**). The selected message will be displayed on your screen.

❹ If you have more than one screenful of headers, type **z+** at the mail prompt (**&**) to move the list forward one screenful; type **z-** to move the list backward one screenful.

❺ To display the screenful of headers that includes a specified message, type **h** *n* at the mail prompt (**&**), where *n* is the message number.

Table 4–1 presents commands and command options for the Mail utility entered at the shell prompt.

Table 4–1 Mail Shell Commands

Command	Options	Action
mail		Without an argument, look for new mail; with a userid and address as arguments, invoke mailer.
mail	**-f ~/mbox**	Reads the *mbox*, where undeleted messages are stored by default.

Table 4–2 presents commands and command options entered at the mail prompt (**&**).

Table 4–2 Mail Commands

Command	Action
co (*n*) *filename*	Copy or append the current or specified message to the named file.
d (*n*)	Delete the current or specified message.
h (*n*)	List message headers; includes the specified message in the listing.
help	Display online help.
n	Display next message.

Table 4–2 (Cont.) Mail Commands

Command	Action
q	Quit ULTRIX Mail; undeleted messages are saved in your *mbox*.
R	Reply to a message.
r	Reply to all users who received the original message.
s (*n*) *filename*	Save current or specified message in the named file.
u (*n*)	Undelete current or specified message.
z+	Move header list forward one screenful.
z-	Move header list back one screenful.

4.7 Forwarding Mail

''Escape commands'' are used to forward mail. Escape commands must be entered on a blank line, and begin with a tilde (~). To forward a message, follow the normal procedure for sending mail, then type a tilde (~) and an escape command on a blank line.

Table 4–3 presents escape commands used to forward mail.

Table 4–3 Mail Escape Commands

Command	Action
~d	Includes *dead.letter* in current message
~f (*msg*)	Includes most recent or specified messages in current message.
~m (*msg*)	Includes most recent or specified messages in current message, indented one tab stop.
~w (*filename*)	Includes named file in current message.

Attention: If you wish to include a tilde (~) as the first character in a line of your message, you must type two tildes (~~).

4.8 Mail Aliases

Mail ''aliases'' are nicknames that users create to simplify long mail addresses. When an alias is entered at the **To:** prompt, it is automatically replaced by the address when the message is sent. Mail aliases are entered in the *.mailrc* file located in your home directory. The following example presents the syntax for mail aliases in the *.mailrc* file.

```
alias boris boris@badinoff.kremlin.edu
alias bullw bullwinkle@frostbite.falls.edu
```

Note: If you do not have a *.mailrc* file, you will find one in the /usr/skel directory. Use the cp command to copy the *.mailrc* file into your home directory, as in the following example:

```
foo(+!) cp /usr/skel/.mailrc ~ RET
```

4.9 Aborting Mail Messages

If you wish to cancel the message you are writing, abort the message by pressing <CTRL/c> twice.

5 GNU Emacs

GNU Emacs is a full-screen, ‘‘self-documenting, customizable, extensible real-time display’’ editor first developed at the Massachusetts Institute of Technology Artificial Intelligence Laboratory by Richard Stallman and Guy Steele. Distributed by the Free Software Foundation (FSF), ‘‘GNU Emacs is *free* ... everyone is free to use it and free to redistribute it on a free basis.’’† This chapter presents summaries of those commands needed most by the new user.‡

5.1 Introduction

Unlike other text editors, Emacs offers users a total computing environment. Countless operations can be performed in multiple windows: files can be edited, copied, renamed and deleted; text can be formatted for text processors; directories can be listed; programs compiled; UNIX shell commands executed. Macros can be created to facilitate the execution of repetitive commands; abbreviations can be written to reduce repetitive editing tasks. Emacs also provides interfaces for electronic mail and UseNet newsreaders.

5.1.1 Emacs Conventions

Emacs employs a variety of command key sequences composed of the Control and ‘‘Meta’’ or Escape keys, plus another key or keys. A number of conventions for these command key sequences have been standardized in Emacs documentation. Table 5–1 presents Emacs command key sequence conventions.

Table 5–1 Emacs Command Conventions

Key	Function
C–	Press the Control key.
DEL	Press the Delete or Backspace key.
M–	Press the Esc or ‘‘Meta’’ key.
RET	Press the Return or Enter key.
SPC	Press the SPACEBAR.
TAB	Press the Tab key.

Note: If your Escape key does not function properly (or if you do not have an Escape key), which sometimes happens when using a modem to connect to CUNYGS or timessqr, use C–[(Control left-bracket).

---

† Richard Stallman, *GNU Emacs Manual*, 3. For more information about GNU Emacs, write to the Free Software Foundation, 1000 Mass Ave. Cambridge, MA 02138.

‡ Future versions of this document will contain a more detailed discussion of the GNU Emacs display editor.

5.2 Getting Started

To access Emacs, type **emacs** at the command prompt and press **RET**. A blank Emacs **scratch* buffer* window will open on your screen. Buffers are temporary storage areas for text. All editing is done in buffers, then saved to files. The Emacs window is composed of four functional areas:

❶ The ‘‘buffer’’ is the area in which you edit text.

❷ The ‘‘point,’’ represented by the ‘‘cursor,’’ marks your position in the buffer.

❸ The ‘‘mode line,’’ located near the bottom of the screen, displays information about your Emacs session, including the buffer name, the name of major and minor modes associated with the buffer, the position of the cursor in the buffer, and the modification status.

❹ The ‘‘minibuffer,’’ also known as the ‘‘echo area,’’ located beneath the mode line, prompts you for commands and reports status information.

5.2.1 Issuing Emacs Commands

The Emacs editor is written in a dialect of the LISP programming language called, not surprisingly, ELISP. Emacs commands are LISP symbols named after functions defined to perform specific tasks. Command key sequences are ‘‘bound’’ to these symbols. Thus to find and read a file in an Emacs buffer, you can type

```
M-x find-file
```

or you can type

```
C-x C-f
```

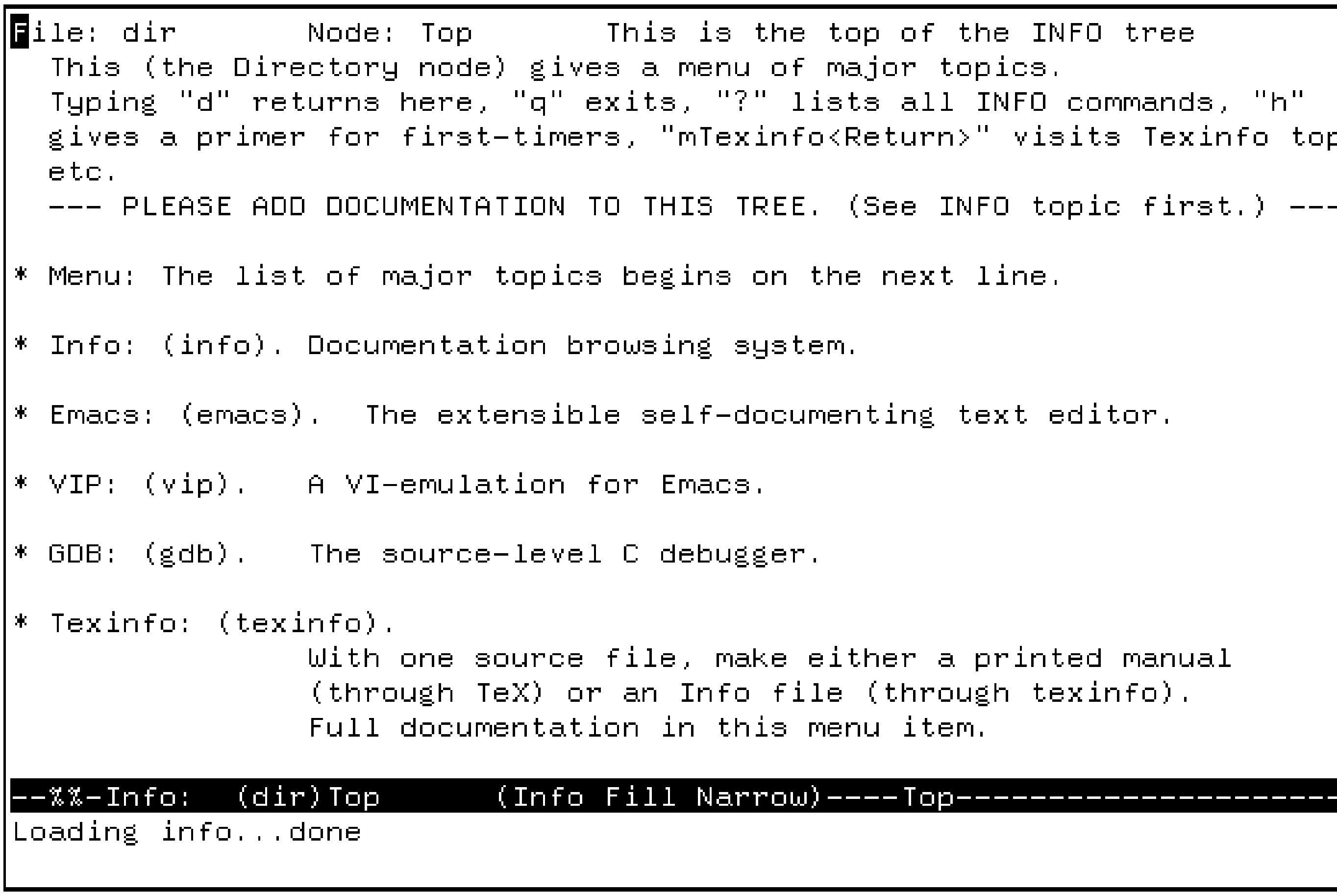
because the command key sequence **C–x C–f** is bound to the **find-file** command. While key bindings can be changed to suit the needs of a particular user or operating system, the function that performs the command remains the same.

To issue an Emacs command using the Control key, press **C–** and the following key *simultaneously*. For example, to issue the ‘‘find-file’’ command (**C–x C–f**), press **C–** and **x f** at the same time.

To issue an Emacs command using the Escape key, press **M–**, *release it*, then press the following key.

‘‘Command line completion’’ is one of the great advantages of GNU Emacs. For example, after typing the first few letters of a filename in response to the **C–x C–f** command (find-file), press TAB. Emacs will complete the filename, create a ‘‘completion list’’ if the letters indicate more than one file, or complain if there is ‘‘No match.’’

Figure 5–1 A GNU Emacs Buffer



```
File: dir       Node: Top       This is the top of the INFO tree
  This (the Directory node) gives a menu of major topics.
  Typing "d" returns here, "q" exits, "?" lists all INFO commands, "h"
  gives a primer for first-timers, "mTexinfo<Return>" visits Texinfo top
  etc.
  --- PLEASE ADD DOCUMENTATION TO THIS TREE. (See INFO topic first.) ---

* Menu: The list of major topics begins on the next line.

* Info: (info). Documentation browsing system.

* Emacs: (emacs).  The extensible self-documenting text editor.

* VIP: (vip).   A VI-emulation for Emacs.

* GDB: (gdb).   The source-level C debugger.

* Texinfo: (texinfo).
              With one source file, make either a printed manual
              (through TeX) or an Info file (through texinfo).
              Full documentation in this menu item.

--%%-Info:  (dir)Top       (Info Fill Narrow)----Top--------------------
Loading info...done
```

5.2.2 The .emacs File

Although Emacs will run properly without an initialization file (often called a ‘‘dotfile’’), it is always better to have a *.emacs* file in your home directory. The Network Operations group has provided an excellent *.emacs* file. To copy this file into your home directory (represented by a tilde), use the following command *as written*:

```
cp /usr/skel/.emacs ~ RET
```

Note: It beyond the scope of this document to explain the significance and the syntax of the *.emacs* file; nevertheless, from time to time I will offer a line of code which, if entered in your .emacs file, will make Emacs easier to use.

5.3 Edit Commands

Table 5–2 lists command key sequences used to move the cursor in buffers.

Table 5–2 Cursor Commands

Sequence	Command	Cursor Movement
C–f	forward-char	Forward one character
C–b	backward-char	Back one character
C–p	previous-line	Up one line
C–n	next-line	Down one line
C–a	beginning-of-line	To beginning of line
C–e	end-of-line	To end of line
C–v	scroll-up	Scroll forward one screen
M–v	scroll-down	Scroll backward one screen
M–f	forward-word	Forward one word
M–b	backward-word	Back one word
M–a	backward-sentence	Back one sentence
M–e	forward-sentence	Forward one sentence
M–(	backward-paragraph	Back one paragraph
M–)	forward-paragraph	Forward one paragraph
M–<	beginning-of-buffer	Back to beginning of buffer
M–>	end-of-buffer	Forward to end of buffer

Table 5–3 presents command key sequences used to manipulate files in buffers.

Table 5–3 File Commands

Sequence	Command	Action
C–l	recenter	Redraw the screen display
C–x C–f	find-file	Display file in buffer
C–x i	insert-file	Insert another file at the cursor point
C–x C–w	write-file	Write contents of buffer to file
C–x C–s	save-buffer	Save buffer contents to file
C–x s	save-some-buffers	Save buffer contents to file
C–x C–c	save-buffers-kill-emacs	Quit Emacs

Note: Depending on your terminal emulation, the standard key binding for the ‘‘save-buffer’’ command (C–x C–s) may not function

properly, because the control key sequence <CTRL/s> is used to freeze the screen. Not to worry. To save the contents of a buffer to a file, use the ''save-some-buffers'' command: first type C–x followed by an s, as listed in the table above. Emacs will ask you to confirm the save in the minibuffer. Type y and the file will be saved to disk.

Table 5–4 presents command key sequences used to manipulate buffers.

Table 5–4 Buffer Commands

Sequence	Command	Action
C–x b	switch-to-buffer	Move to specified buffer
C–x o	other-window	Move cursor clockwise to another window
C–x 4 b	switch-to-buffer-other-window	Move to buffer in another window
C–x C–b	list-buffers	Display buffer list
C–x k	kill-buffer	Delete specified buffer
C–x 0	delete-window	Delete current window
C–x 1	delete-other-windows	Delete all but current window

Table 5–5 presents command key sequences used to format text.

Table 5–5 Formatting Commands

Sequence	Command	Action
M–q	fill-paragraph	Format paragraph
M–g	fill-region	Format region
C–u M–q		Justify paragraph

5.3.1 Search and Replace

Emacs offers a variety of methods to search and replace text, including simple searches of words and phrases, ''incremental'' searches, which begin with the first character of the search text, ''regular expression'' or ''pattern matching'' searches, and ''query-replace'' searches, which request confirmation before replacing the found text with new text specified by the user.

Table 5–6 presents command key sequences used to search for character strings in buffers.

Table 5–6 Search Commands

Sequence	Command	Action
C–s Esc C–w	word-search-forward	Search forward for word

Table 5–6 (Cont.) Search Commands

Sequence	Command	Action
C–r Esc C–w	word-search-backward	Search backward for word
C–r	isearch-backward	Search backward to beginning of buffer
C–s	isearch-forward	Search forward to end of buffer
M–%	query-replace	Search forward and wait for confirmation of replacement

As stated above, **C–s** is used to freeze the screen. The most commonly used search command is ‘‘isearch-forward’’, bound to **C–s**. To avoid this problem, use the command name when specifying a forward search, as follows:

```
M–x isearch–forward
```

If typing ‘‘**M–x isearch-forward**’’ seems tedious, there is a more efficient way to perform a forward incremental search: create a key binding for the command. This is done by adding a line of ELISP code in your *.emacs* file. For example, if you would like to bind **M–s** to the ‘‘isearch-forward’’ command, insert the following line *as written* in your *.emacs* file:

```
(define-key global-map "\M-s" 'isearch-forward)
```

5.3.2 Cut and Paste

To ‘‘cut and paste’’ in Emacs parlance is to ‘‘kill and yank’’. Text can be deleted by character, word, line, region, and buffer. (A ‘‘region’’ of text is a section ‘‘marked’’ or selected for various operations, such as deletion.) Deleted words, lines, regions, and buffers are placed in a ‘‘kill ring,’’ where they can be recovered with a ‘‘yank’’ command.

To mark text for deletion, place the cursor at the beginning of the text you wish to delete and enter the **C–SPC** or **C–@** command. The message, ‘‘Mark set,’’ will appear in the minibuffer. Move the cursor to the end of the text to be deleted and enter the **C–w** command. The text region will disappear into the kill ring, where it will stay until the kill ring is full, or until you end your Emacs session. If you wish to yank the deleted text elsewhere in the current buffer, move the cursor to the desired point and enter the **C–y** command. To yank the text in another file, open the file with the **C–x f** command and yank the text with **C–y**.

By default, the kill ring contains 30 deletions. The ‘‘yank-pop’’ command (**M–y**) is used to retrieve those deletions from the kill ring. **M–y** replaces the most recent yank with the yank that precedes it in the kill ring. The ‘‘yank-pop’’ command can be repeated in order to move through the kill ring. **M–y** is helpful when trying to choose between adjectives; for example, if you have deleted ‘‘dark and stormy’’ from the phrase, ‘‘It was a dark and stormy night,’’ and replaced it with ‘‘hot and steamy,’’ then replaced that with ‘‘cool and breezy,’’ you can use the ‘‘yank-pop’’ command to repeat each adjectival phrase in sequence until you find the one that best expresses your idea.

Note: Marked regions are invisible; other than the ‘‘Mark set’’ message in the minibuffer, there is no indication that a region has been marked. If you would like to verify the marked region, use the C–x C–x command, ‘‘exchange-point-and-mark.’’ The cursor will move to the beginning of the region. Repeat the C–x C–x command and the cursor will move back to the end of the region.

Table 5–7 presents command key sequences used to mark, kill, and yank text.

Table 5–7 Kill and Yank Commands

Sequence	Command	Action
C–x u	advertised-undo	Undo last edit command
C–g	keyboard-quit	Cancel current command
DEL	delete-backward-char	Delete previous character
C–d	delete-char	Delete the character under the cursor
M–d	kill-word	Delete the word under the cursor
C–k	kill-line	Delete from the cursor to line′s end
C–w	kill-region	Delete marked region
M–w	copy-region-as-kill	Copy marked region into kill ring
C–y	yank	Paste deleted text
M–y	yank-pop	Paste most recent deletion
C–SPC	set-mark-command	Mark beginning or end of region
C–@	set-mark-command	Mark beginning or end of region
M–h	mark-paragraph	Mark a paragraph
C–x h	mark-whole-buffer	Mark entire buffer
C–x C–x	exchange-point-and-mark	Move cursor to begining or end of region

5.4 Emacs Mail and RMAIL

Emacs offers users very powerful and convenient electronic mail interfaces. The Mail program is used to write and send mail messages; RMAIL is used to read and store mail messages.

Emacs Mail recognizes aliases defined in the *.mailrc* file.

5.4.1 Mail

To invoke Emacs Mail, type **C–x m** and press RET. The Mail window will appear in your Emacs buffer containing the **To:** and **Subject:** fields. The cursor will be placed after the **To:** field. Type the address of the recipient. Multiple addresses can also be entered at the **To:** field, provided that a comma (,) separates each address. Using the down arrow key or the **C–n** command, move the cursor to the **Subject:** field. Enter a subject if you so desire, then move the cursor to the message area:

```
To: boris@badinoff.kremlin.edu
Subject: Capitalism
--text follows this line--
   .
   .
   .
```

Edit your message with standard Emacs commands. When you have completed your message, enter the **C–c C–c** command to send it. Emacs Mail will confirm the send with the message ‘‘Sending...done’’ in the minibuffer.

The **C–c C–f C–c** command is used to create a **CC:** field for addresses to which copies of the message are to be sent. If several copies are to be sent, separate each address with a comma (,). If the addresses do not fit on one line, repeat the **C–c C–f C–c** to create another **CC:** field.

If you have a *.signature* file in your home directory that you would like to include in your message, enter the **C–c C–w** command.

Table 5–8 presents command key sequences used to edit and send mail messages.

Table 5–8 Mail Commands

Sequence	Command	Action
C–x m	mail	Invoke Emacs Mail
C–x 4 m	mail-other-window	Invoke Mail in another window
C–c C–f C–t	mail-to	Create and move to the To: field
C–c C–f C–c	mail-cc	Create and move to the CC: field for sending a copy
C–c C–f C–s	mail-subject	Create and move to the Subject: field

Table 5–8 (Cont.) Mail Commands

Sequence	Command	Action
C–c C–w	mail-signature	Insert the .signature file
C–x i *filename*	insert-file	Insert a file to be mailed
C–c C–c	mail-send-and-exit	Send the message and exit Mail
C–c C–s	mail-send	Send the message without exiting Mail

5.4.2 RMAIL

The RMAIL program is used to read and manage mail messages. When mail arrives, it is stored in a system mailbox. When RMAIL is invoked, all incoming messages are moved from the system mailbox to an inbox in the user's home directory called, not surprisingly, *RMAIL.*

5.4.2.1 Invoking RMAIL

To invoke RMAIL, type **M–x rmail** and press RET. RMAIL will respond with the first unread message, or the most recent read message, in your inbox.

Most RMAIL commands consist of a single letter, as listed in the table below.

5.4.2.2 Listing RMAIL Messages

If you have received several messages, you can list them with the **h** command (short for ''header''). RMAIL will open two windows in the buffer: the ''RMAIL-summary'' window, containing the header list; and the RMAIL window, containing the last read message or the first unread message. As you move up and down the list with the arrow keys or the **p** and **n** commands, RMAIL will display the indicated message in the RMAIL window.

5.4.2.3 Reading RMAIL Messages

The spacebar (SPC) is used to scroll forward in a message; the delete key (DEL) is used to scroll backward. The **n** command is used to move to the next message; the **p** command is used to move to the previous message. Note: The **n** and **p** commands skip messages marked for deletion. To move forward or backward to marked messages, use the **M–n** and **M–p** commands.

5.4.2.4 Deleting and "Undeleting" RMAIL Messages

Messages are marked for deletion with the **d** command, and subsequently deleted when you exit RMAIL. The **u** command is used to ''undelete'' messages marked for deletion.

5.4.2.5 Saving RMAIL Messages

Messages that are not deleted remain in the *RMAIL* file unless they are saved to another RMAIL format file with the **o** command, or to an ASCII text file with the **C–o** command. The difference between these two file formats is significant if you intend to move your mail files to another computer or to read your mail with a different Mail User Agent (MUA). Use the **o** command if you intend to read your mail exclusively in RMAIL.

5.4.2.6 Forwarding and Replying to Messages

Mail is forwarded with the **f** command. Use the **r** command to reply to a mail message. These commands place an Emacs Mail buffer on top of the RMAIL buffer, allowing you to edit the message to be forwarded or, if you are composing a reply, to insert the original message with the **C–c C–y**.

5.4.2.7 Saving Copies of Mail

Although the easiest way to maintain a record of the messages you send is to send yourself a copy, the most efficient method is to have the copy sent to a file in your home directory. For this you must set the ‘‘mail-archive-file-name’’ variable in your *.emacs* file. Using the Emacs editor, append the following line of ELISP code to your *.emacs* file:

```
(setq mail-archive-file-name "~/outbox")
```

Note that any filename can be specified, provided that the syntax is correct.

Table 5–9 presents command key sequences used to read, save, reply to and forward mail messages.

Table 5–9 RMAIL Commands

Key Command	Action
M–x rmail	Begin RMAIL
SPC	Scroll message forward
DEL	Scroll message backward
.	Move to the top of the message
j	Jump to first message
n	Move to next message
p	Move to previous message
>	Move to last message
g	Check for new messages
d	Mark message for deletion and move forward
C–d	Mark message for deletion and move backward
M–n	Move to next message; disregard deletion mark
M–p	Move to previous message; disregard deletion mark
u	Undelete message marked for deletion

Table 5–9 (Cont.) RMAIL Commands

Key Command	Action
x	Delete all marked messages
e	Delete all marked messages
m	Invoke Emacs Mail in RMAIL
r	Reply to current message
C–c C–y	Include current message in reply
f	Forward current message
o *filename*	Save current message in RMAIL format
C–o *filename*	Save current message in UNIX format
M–s	Search for regular expression
h	Show list of messages by header

5.5 GNUS

GNUS, a subsystem of Emacs, is the ultimate UseNet newsreader. Like Emacs, GNUS offers command line completion and completion lists. Like RMAIL, GNUS displays two windows: one containing a list of subject headers; the other containing the current article.

Before invoking GNUS, you must enter a few lines of ELISP code into your *.emacs* file. The Network Operations group has in fact created a *.emacs* file containing this code. If you have not modified your *.emacs* file, you will find the NetOps *.emacs* file in **timessqr**'s /u/questions directory. If you have modified your *.emacs* file, append the following code to it, *as written*:

```
(autoload ’gnus "gnus" "Read Netnews." t)
(autoload ’gnus-post-news "gnus" "Post Netnews." t)
(setq gnus-nntp-service nil)
(setq nnspool-inews-program "/usr/local/bin/inews")
(setq nnspool-active-file "/usr/users/news/active")
(setq gnus-your-domain "gc.cuny.edu")
(setq gnus-your-organization "Graduate School and University Center, C.U.N.Y, New York")
(setq gnus-novice-user t)  ;; prompts when posting an article
```

Most GNUS commands consist of a single letter. Often the same letter command is used in each of the three GNUS buffers: Newsgroup, where newsgroups are listed; Subject, where subject headers are listed; and Article, where the article is displayed. For example, in the Newsgroup buffer, the spacebar (SPC) is used to display a selected newsgroup's subject header list; in the Subject buffer, SPC is used to display an article from the subject header list; and in the Article buffer, SPC is used to scroll forward in an article. The **q** command is used to quit both the Subject and Newsgroup buffers.

To invoke GNUS, type **M–x gnus** and press RET. You will see a series of brief messages in the minibuffer, ending with

```
NNTP server:
```

Press RET. If you have never used GNUS before, GNUS will automatically subscribe you to all available newsgroups. Finally the Newsgroup buffer will appear:

```
  9497: rec.arts.drwho
   428: comp.unix.large
  3260: comp.unix.internals
   739: comp.lang.objective-c
  3705: comp.fonts
   181: rec.games.mud.misc
  5052: comp.binaries.ibm.pc.archives
   395: comp.research.japan
  1330: talk.rumors
  3637: sci.virtual-worlds
 29894: rec.bicycles
  8617: comp.unix.ultrix
 21935: comp.unix.sysv386
 17564: comp.unix.questions
  .
  .
  .
```

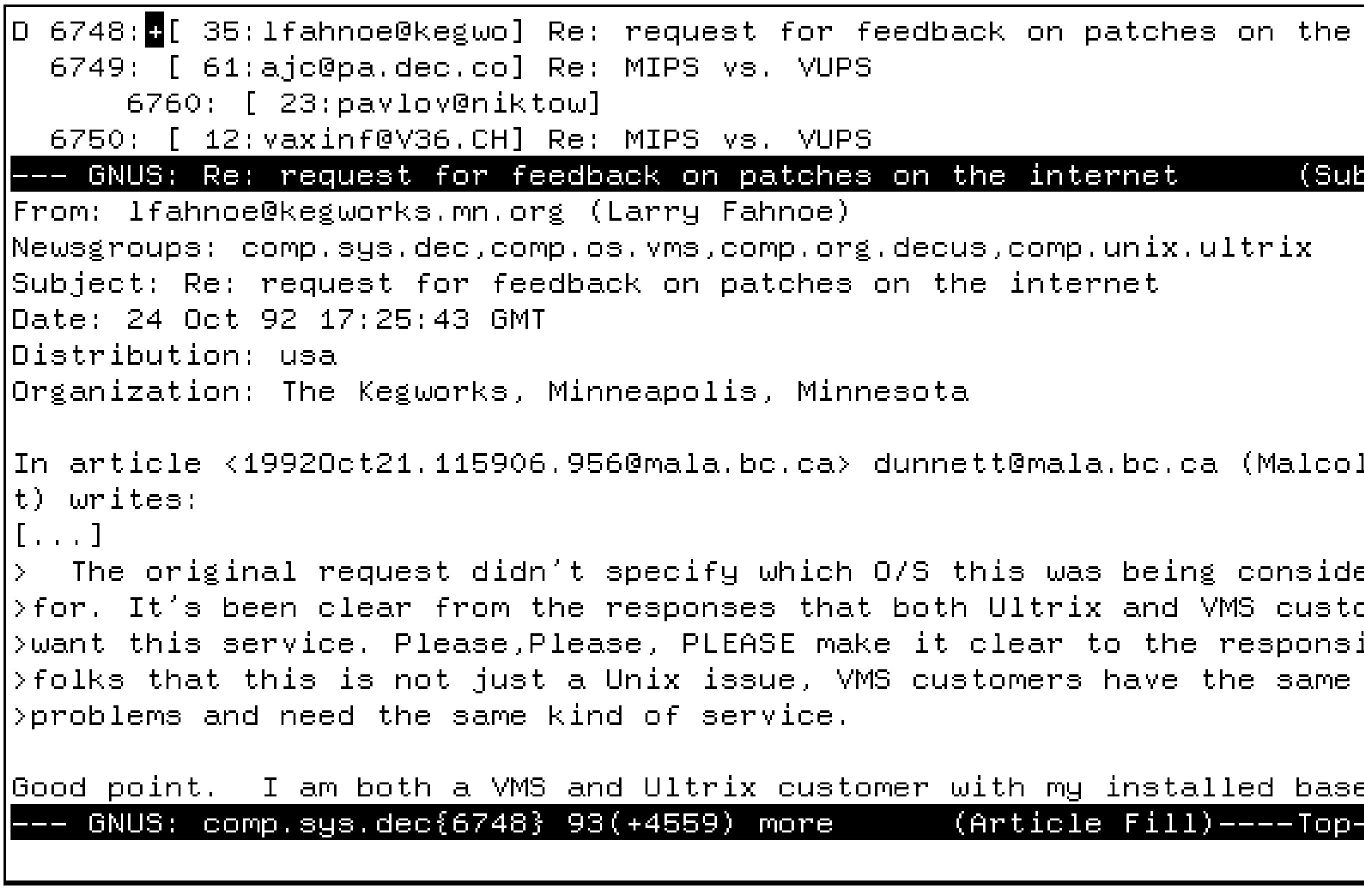
Use the **j** command to ‘‘jump’’ to a newsgroup; you will be prompted for a newsgroup in the minibuffer. Type the newsgroup’s name, press RET, and the cursor will find its way to the selected newsgroup. Press SPC. The total number of articles in the newsgroup will appear in the minibuffer, and you will be asked:

```
How many articles from comp.unix.ultrix (default 8663):
```

If you press return, GNUS will attempt to list all of the articles in the newsgroup. *Pick a reasonable number of articles.* Do not just press RET, because GNUS will force you to wait while it spends time looking for more articles than you have time to read. Eventually two windows will appear: the top window containing the Subject buffer’s list of subject headers; and the bottom window containing the first of your selected articles. Note the capital ‘‘D’’ in the left column of the Subject buffer: each article you read is marked for deletion. When you quit GNUS, marked articles are deleted from the Subject header list. (GNUS maintains a record of your newsgroups and available articles in the *.newsrc* file; this file is created automatically in your home directory the first time you invoke GNUS.)

Table 5–10 lists GNUS Newsgroup buffer commands.

Figure 5–2 A GNUS Buffer



```
D 6748:+[ 35:lfahnoe@kegwo] Re: request for feedback on patches on the
  6749: [ 61:ajc@pa.dec.co] Re: MIPS vs. VUPS
      6760: [ 23:pavlov@niktow]
  6750: [ 12:vaxinf@V36.CH] Re: MIPS vs. VUPS
--- GNUS: Re: request for feedback on patches on the internet      (Sub
From: lfahnoe@kegworks.mn.org (Larry Fahnoe)
Newsgroups: comp.sys.dec,comp.os.vms,comp.org.decus,comp.unix.ultrix
Subject: Re: request for feedback on patches on the internet
Date: 24 Oct 92 17:25:43 GMT
Distribution: usa
Organization: The Kegworks, Minneapolis, Minnesota

In article <1992Oct21.115906.956@mala.bc.ca> dunnett@mala.bc.ca (Malcol
t) writes:
[...]
>  The original request didn't specify which O/S this was being conside
>for. It's been clear from the responses that both Ultrix and VMS custo
>want this service. Please,Please, PLEASE make it clear to the responsi
>folks that this is not just a Unix issue, VMS customers have the same
>problems and need the same kind of service.

Good point.  I am both a VMS and Ultrix customer with my installed base
--- GNUS: comp.sys.dec{6748} 93(+4559) more      (Article Fill)----Top-
```

Table 5–10 GNUS Newsgroup Commands

Key	Command	Action
n	gnus-Group-next-unread-group	Move to next newsgroup of unread articles
p	gnus-Group-next-unread-group	Move to previous newsgroup of unread articles
N	gnus-Group-next-group	Move to next newsgroup
C–n	gnus-Group-next-group	Move to next newsgroup
P	gnus-Group-prev-group	Move to previous newsgroup
C–p	gnus-Group-prev-group	Move to previous newsgroup
j	gnus-Group-jump-to-group	Move to a specified newsgroup
/	isearch-forward	Incremental search forward
<	beginning-of-buffer	Move to beginning of Newsgroup buffer
>	end-of-buffer	Move to end of Newsgroup buffer
SPC	gnus-Group-read-group	Select newsgroup from list

Table 5–11 lists GNUS Subject buffer commands.

Table 5–11 GNUS Subject Commands

Key	Command	Action
C–n	gnus-Subject-next-subject	Move to next header
C–p	gnus-Subject-prev-subject	Move to previous header
M–n	gnus-Subject-next-unread-subject	Move to next unmarked header
M–p	gnus-Subject-prev-unread-subject	Move to previous unmarked header
j *number* RET	gnus-Subject-goto-subject	Move to header specified by header number

Table 5–11 (Cont.) GNUS Subject Commands

Key	Command	Action
/	isearch-forward	Incremental search of headers
SPC	gnus-Subject-next-page	Select article
g	gnus-Subject-show-article	Select article
n	gnus-Subject-next-unread-article	Read next unmarked article
p	gnus-Subject-prev-unread-article	Read previous unmarked article
N	gnus-Subject-next-article	Read next article
P	gnus-Subject-prev-article	Read previous article
M–C–n	gnus-Subject-next-same-subject	Read next article with the same subject (follow a \`\`thread'')
M–C–p	gnus-Subject-prev-same-subject	Read previous article with same subject (follow a \`\`thread'')
.	gnus-Subject-first-unread-article	Read first unread article
l	gnus-Subject-goto-last-article	Read the last selected article
J *number* RET	gnus-Subject-goto-article	Read article specified by article number
C–c C–s n	gnus-Subject-sort-by-number	Sort headers by number
C–c C–s s	gnus-Subject-sort-by-subject	Sort headers by subject
C–c C–s d	gnus-Subject-sort-by-date	Sort headers by date
C–c C–s a	gnus-Subject-sort-by-author	Sort headers by auther

Table 5–12 lists GNUS Article buffer commands.

Table 5–12 GNUS Article Commands

Key	Command	Action
SPC	gnus-Subject-next-page	Scroll down article
DEL	gnus-Subject-prev-page	Scroll up article
<	gnus-Subject-beginning-of-article	Move to top of article

Table 5–12 (Cont.) GNUS Article Commands

Key	Command	Action
>	gnus-Subject-end-of-article	Move to end of article
v	gnus-Subject-show-all-headers	Show all article headers
u	gnus-Subject-mark-as-unread-forward	Mark article as unread and move to next subject
M–u	gnus-Subject-clear-mark-forward	Clear marks and move to next subject
o	gnus-Subject-save-article	Save article in RMAIL format.
C–o	gnus-Subject-save-in-rmail	Save article in UNIX mailbox format

When saving articles, GNUS will prompt you for a filename. You can save several articles to the same file.

Table 5–13 presents commands used to post articles to newsgroups.

Table 5–13 GNUS Post Commands

Key	Command	Action
f	gnus-Subject-post-reply	Post followup to current article
F	gnus-Subject-post-reply-with-original	Include current article in reply
a	gnus-Subject-post-news	Post a new article
M–x postnews		Post a new article
C	gnus-Subject-cancel-article	Cancel your posted article
r	gnus-Subject-mail-reply	Reply to article's author in mail
R	gnus-Subject-mail-reply-with-original	Include current article in reply to author
m	gnus-Subject-mail-other-window	Edit a mail message in another window

To quit GNUS, move the cursor to the Subject buffer and type **q**. When the Newsgroup buffer reappears, type **q** again. GNUS will ask you if you really want to quit. Type **y**. GNUS will update your *.newsrc* file and quit.

5.6 Emacs Help

Emacs offers extensive online help, including an excellent tutorial.

Table 5–14 Help Command Key Sequences

Sequence	Command	Action
C–h	help-command	Begin help command
C–h i	info	Display INFO menu
C–h ?	help-for-help	Describe help command
C–h t	help-with-tutorial	Begin interactive tutorial
M–x doctor		Call online psychoanalyst

5.7 Emacs Documentation

A number of fine manuals about GNU Emacs are commercially available, some of which are listed here:

❶ Richard Stallman's *GNU Emacs Manual* (Free Software Foundation, 1986)

❷ *Learning GNU Emacs* (O'Reilly & Associates, 1992)

❸ *GNU Emacs: UNIX Text Editing and Programming* (Hewlett Packard, 1992)

6 Coming Attractions

Topics to be included in future versions of this document include:

❶ Customizing Your Computing Environment

❷ The ‘‘archie’’ database.

❸ Anonymous *ftp*.

❹ Process management for group research.

❺ Suggestions are welcome.

The author wishes to acknowledge a number of individuals who prefer to remain anonymous.

A Pine-Mail

Pine is a MIME-capable Mail User Agent. MIME, an acronym for ‘‘Multipurpose Internet Mail Extensions,’’ permits the inclusion of binary data in Internet mail messages. With Pine, users are no longer limited to the ASCII text format: documents produced with word processors, spreadsheets, images, audio, and other binary data can be included in mail messages.

To access Pine, type **pine-mail** at the shell prompt and press RET. The Pine display will appear, as follows:

```
foo(+!) pine-mail RET
        PINE 3.05        MAIN MENU        Folder:inbox  0 Messages


      ?   HELP        - Get help using Pine

      C   COMPOSE     - Compose and send a message

      I   MAIL INDEX  - Read mail in current folder

      F   FOLDERS     - Open a different mail folder

      A   ADDRESSES   - Update your address book

      O   OTHER       - Use other functions

      Q   QUIT        - Exit the Pine mail program


                    [Folder "inbox" opened with 0 messages]
? Help       Q Quit       F Folders     O Other
C Compose    I Mail Index A Addresses
```

To use Pine, choose a command from the menu and follow the displayed instructions.

B Gopher

The Gopher Information Server provides easy access to online databases, libraries, and computer systems located all over the world.

To invoke the Gopher Information Server, type **gopher** at the shell prompt and press RET, as in the following example:

```
foo(+!) gopher RET
                    Internet Gopher Information Client v1.02

                    Root gopher server: timessqr.gc.cuny.edu

 -->  1.  About CUNY Gopher Server.
      2.  About Graduate School at CUNY/
      3.  About New York City/
      4.  Documentation Online/
      5.  Finding People/
      6.  Libraries and Related Stuff/
      7.  Lots of interesting stuff (from MSEN Inc)/
      8.  Other Gopher and Information Servers/
      9.  Weather/


Press ? for Help, q to Quit, u to go up a menu                     Page: 1/1
```

Gopher directories are chosen with the up and down arrow keys. Note the instructions at the botton of the display. In the above example, the root directory, ‘‘About CUNY Gopher Server,’’ is chosen. Press RET and the following screen will be displayed:

```
Last update: 10/31/92          Next update: 11/92

 Welcome to the Graduate School and University Center of CUNY
                          GOPHER SERVER
                [SAVE A TREE FEED A GOPHER PROJECT]

A "gopher" is a distributed document delivery service.  It allows a
neophyte user to access various types of data residing on multiple
hosts spread over many domains spanning countries and continents in a
seamless fashion.  This is accomplished by presenting the user a
hierarchical arrangment of documents and by using a client-server
communications model.

Gopher code and documentation is distributed by U of Minnesota.
Obtain by using Gopher or anonymous FTP from:

        boombox.micro.umn.edu  in /pub/gopher

The information database for the Graduate Center ( short for Graduate
School and University Center of City University of New York) server is
in its infancy now, and should be considered experimental
during this process. If you want any information service or have

--More--(61%)[Press space to continue, 'q' to quit.]
```

Press the spacebar (SPC) and the next screen will appear:

```
suggestions please feel free to send email to

 gopher@netops.gc.cuny.edu
 gopher-master@gopher.gc.cuny.edu

Under New York City: Bookstores Guide
   Theatre Guide
   Hotel Information
   Facts about New York City

This gopher is enhanced and maintained by the Network and Operations
Staff during off-hours.  The work on this project is totally on
volunteered time -- mostly on Saturdays.

Interested people are welcome to drop by and chat about this and other
projects with the staff after 5:00 PM on week-days and on Saturdays in
either

 Rm 302A, 302B or Rm 303 in the Computer Center.


Press RETURN to continue, s to save, or p to print:
```

C Accessing timessqr from the Grace Bldg.

The Computer Center Network Operations group provides high speed modem access to **timessqr** for PC users located in the Grace building. ProComm, a menu-driven telecommunications software, is installed with the System 85 access service. The following code example presents a command file users can use to facilitate access to **timessqr**.

```
EMULATE VT100
SET BAUDRATE 19200
SET PARITY EVEN
SET DATABITS 7
SET DUPLEX HALF
BACK:
BREAK
BREAK
PAUSE 1
TRANSMIT "^M"
WAITFOR "DIAL:" 3
IF NOT WAITFOR
GOTO BACK
ENDIF
TRANSMIT "5050!"
SET PARITY NONE
SET DATABITS 8
SET BAUDRATE 19200
SET DUPLEX FULL
WAITFOR "LOCAL>" 3
TRANSMIT "^M"
WAITFOR "Local>"
TRANSMIT "C TIMESSQR^M"
```

To use this command file:

❶ Using the DOS editor or a word processor, copy this command file *exactly as written*.

❷ Name the file *ULTRIX5050.CMD*.

❸ Save the file as an ASCII text.

❹ Copy the file into the **PROCOMM** directory located on the **C:** drive.

❺ Move to the **PROCOMM** directory.

❻ Type **procomm /fULTRIX5050.CMD** to access **timessqr**, as follows:

```
C:\> cd procomm RET
C:\PROCOMM\> procomm /fULTRIX5050.CMD RET
```

Note: Press ALT F5 for a list of available command files.