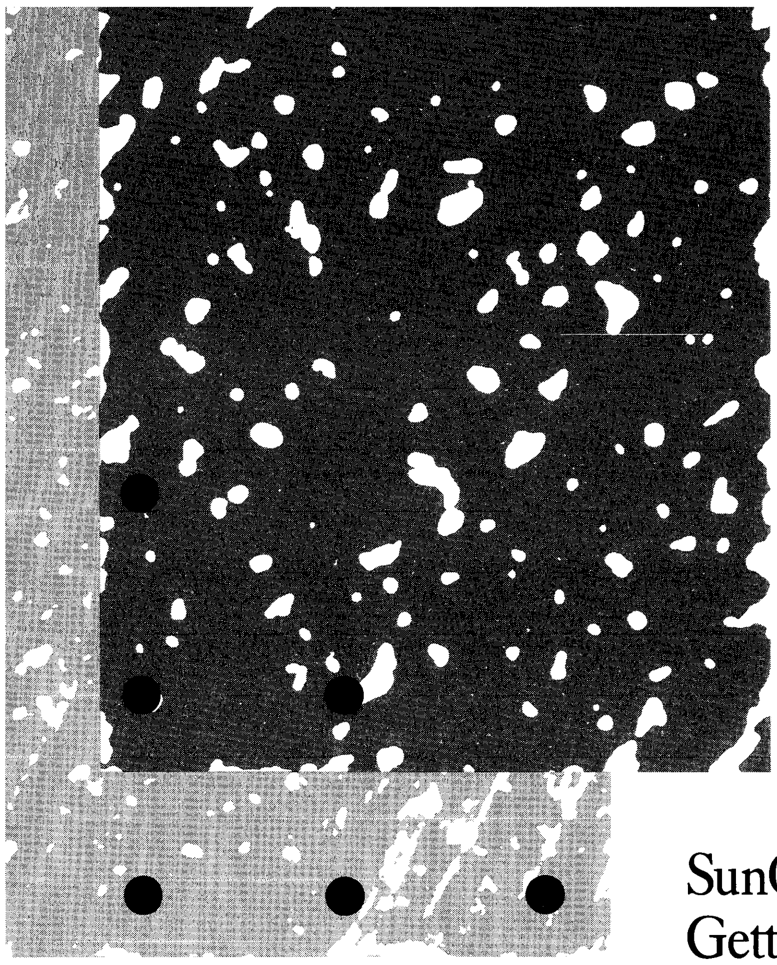


SunOS User's Guide: Getting Started

Contents

Tables

Figures

Preface

SunOS User's Guide: Getting Started introduces the basic knowledge you need to use the SunOS™ operating system and the Sun Workstation®. It assumes that you are new to the Sun Workstation and have little or no experience with the operating system. This manual provides tutorials, not detailed explanations of the inner workings of commands and programs.

SunOS User's Guide: Getting Started introduces

- Special characteristics of the keyboard.
- The essential concepts of the SunOS file system.
- The basic commands for working with files.
- Use of the `vi` text editor.
- The `nroff` and `troff` text formatters.
- Use of the SunOS C shell command interpreter.
- Commands and concepts associated with using the network.
- Sending and receiving messages and mail.

Not covered in this manual is the use of windows systems. There is, however, a chapter on the SunView™ Mail Tool.

Companion Documents

Other manuals in this series cover more specialized topics. We suggest you read them in the following order: *SunView User's Guide* (if your system is running SunView), *SunOS User's Guide: Customizing Your Environment*, *SunOS User's Guide: Doing More*, *SunOS User's Guide: Basic Troubleshooting*.

Book Conventions

Italic font is used for emphasis and for variables that you replace with a specific word or string.

Bold font is used for SunView buttons.

[Keys] on the keyboard are shown this way.

`Typewriter` font is used for what you see on the screen or for file names.

`Bold typewriter` font is represents text you type exactly as shown.

1

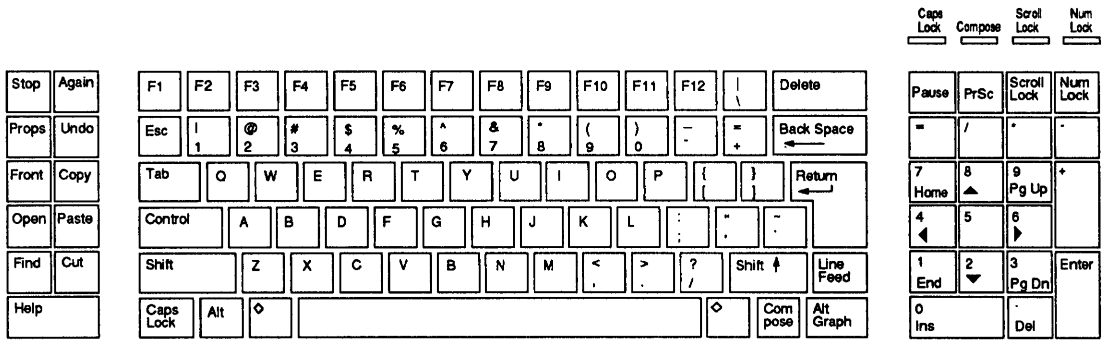
The Keyboard

In this chapter you'll learn some of the special features of the Sun Workstation keyboard and how it differs from a standard typewriter keyboard. Control keys and escape keys, which extend the keyboard's functionality, are introduced.

1.1. Keyboard Layout

The keyboard of a Sun Workstation is like the keyboard of a standard typewriter, with the addition of some special keys.

Figure 1-1 *The Type 4 Keyboard*



See the *SunView User's Guide* for more information on the use of windows-related keys.

NOTE *Remember that the SunOS operating system distinguishes between lowercase and uppercase characters and that it generally prefers lowercase.*

The Return Key

$\boxed{\text{Return}}$ makes the system interpret a command line and execute it. Until you type $\boxed{\text{Return}}$ you can correct the command line.

The Space Bar and the Tab Key

Use the [Space Bar] to add spaces to the command line when necessary.

The [Tab] key inserts up to eight spaces until the next tab stop. Tab stops are regular divisions of the input line.

The Delete and Back Space Keys

Depending on your system, either the [Del] (delete) or the [Back Space] key allows you to back up one character and "rub out" or erase characters in the commands you type.

To correct typing mistakes, rub out characters and then retype. You can rub out characters only back to the beginning of a command line.

The SunOS operating system allows you to change the functions of these and other keys. See *SunOS User's Guide: Customizing Your Environment* for more information.

Control Keys

Control keys are keys that require you to hold down [Ctrl] while typing another key.

[Ctrl-U], the *line-kill character*, erases the entire command line. For example, to erase the entire `passwd` command, hold down the [Ctrl] key and type [u] or [U]:

Before:

```
venus% pas sdw
```

After:

```
venus%
```

[Ctrl-U] doesn't appear on the screen when you type it. Some control keys appear on the screen; others don't.

Table 1-1 *Some Useful Control Keys*

A *word* is a sequence of characters surrounded by space(s) and/or tab(s).

Key	*Appearance*	*Function*
Ctrl-U	*invisible*	Erases entire command line; the *kill character*
Ctrl-W	*invisible*	Erases last *word* on command line
Ctrl-C	`^C`	*Interrupts* many programs and shell scripts
Ctrl-Z	`^Z`	*Suspends* many programs and shell scripts
Ctrl-S	*invisible*	Stops output of running program; prevents output from running off end of screen.
Ctrl-Q	*invisible*	Resumes output from program stopped by Ctrl-S
Ctrl-O	`^O`	Throws away output from program without interrupting the program
Ctrl-D	`^D`	End-of-file character used for logout; also terminates file input
Ctrl-\	`^\`	*Quits* program and saves image of program in file called `core` for later debugging

Escape Keys or Meta-Keys

In contrast to control keys, escape keys require that you type the Esc key, *release* it, and then type the complementing key.

For example, to type Esc-U, press the Esc key, release it, and then type u.

2 A Work Session

Before you start your first SunOS work session, make sure that your system hardware is set up and that you have an account on your system. (Your system adminstrator should have arranged this for you.)

This chapter will teach you how to

- Log in to the machine,
- Set and change your password, and
- Log out.

Turn on the workstation and use it to while you work through this tutorial. If you get stuck, try asking a regular user of the system for help. Or look in *SunOS User's Guide: Basic Troubleshooting*.

Don't be afraid to play with the computer: the best way to learn is to dig in and try the examples.

2.1. Logging In

Your workstation screen will look something like this:

```
venus login%
```

This is called the *system login prompt*. Usually, the word before `login:` is the name of your machine. In this example, the machine name (also called *hostname*) is `venus`.

Since the SunOS operating system will allow more than one person to use the system at the same time, it requires you to identify yourself.

When you get your account on the system, remember your *username* and *password*. Your username (also known as a *login name*, a *login id*, a *user id*, or an *account*) identifies you to the system and to other users. Your password restricts use of your account to those people who know the password. You'll learn how to change your password in Section 2.2 .

- At the login prompt, type your username, followed by [Return].

If you make a mistake in typing a character of your username or password, use the [Del] key to delete the character, and then type the correct character.

```
venus login: medici
Password:
```

The username is `medici` in this example.

The system does not do anything until you complete the command by typing [Return]. After it accepts your username, it prompts you for your password. (If your account does not have a password assigned to it, the system will log you in without asking for a password.)

- Type your password at the password prompt.

  The system does not *echo* (type out) your password on the screen. This prevents other users discovering your password.

- Type [Return] after your password, so the system will interpret what you typed. (From now on, type [Return] after each command.)

If you mistype your username or your password, the system responds with:

```
venus login: mdci
Password:
Login incorrect.
login:
```

Simply retype your username and password to log in. (If the system persists in refusing to log you in, your account may not be set up properly. Try talking with your system administrator or look in the *System and Network Administration* manual.)

After you correctly type your username and password, the system will pause briefly and then type out something like this:

```
venus login: medici
Password:
Last login: Fri Oct 31 23:59:59 from console
SunOS Release 4.1 (DIONE_CLIENT) #1: Fri Feb 14 00:
venus%
```

In addition, the system may type a login message that the operator has entered on the system to keep you informed about important system events, such as "downtime," when the system is shut down, usually for system maintenance. All the information that the system types just after you log in is known as the "message of the day."

The system may also inform you that

```
You have mail.
```

For information on how to read your mail, see Chapters 10 and 11.

The last line in this example is the *command prompt*, consisting of the machine name and a percent sign (`%`).

2.2. Changing Your Password

Right after logging in for the first time, you should change your password: this will ensure that you are the only user with easy access to your account. In the interests of your own security, change your password immediately if you believe someone has used your account without your permission.

For more on security, see *SunOS User's Guide: Doing More.*

Your personal password is your choice entirely. Pick a password that you can remember without writing it down. For security's sake, pick a password that is at least six characters long, has at least one number in it, and is not obvious or easy to guess. Do not use your own name or the name of your spouse. Do not use the names of pets or common objects.

Then type the `passwd` command.

```
venus% passwd
Changing password for medici on venus
Old password:
New password:
Retype new password:
venus%
```

- When the system prompts you for

  `Old password:`,

type in your current password. (If no password was assigned to your account, the system will skip the `Old Password:` prompt.) Just as during log-in, the system will not display passwords.

- Now type in the new password at the next system prompt:

  `New password:`

  The password does not echo. The system asks you to retype your new password to verify spelling.

  If you don't remember or mistype your old password, the system refuses to change your password and responds with

  `Sorry.`

  If this happens repeatedly, contact your system administrator to get a new password.

If you pick a password less than six characters long, the system does not allow you to enter it.

Password Aging

If your system is using the *password aging* process, your password may have a maximum or a maximum *and* minimum lifespan. This could mean that your password will have to be changed by a certain date or cannot be changed until after a certain date. Maximum and minimum lifespans are set by your system administrator. See *SunOS User's Guide: Doing More* and *System and Network Administration* for more information.

2.3. Logging Out

When you've finished your work session, you should secure your screen to protect the integrity of your files. Logging out, which ends a work session, is one way to do this.

If you're running SunView, you can *lock* your screen so that others cannot use your terminal. You can also exit SunView and logout. See *SunView User's Guide.*

Log out using the `logout` command:

```
venus% logout

venus login:
```

Another way to log out is to type `Ctrl-D`, the *end-of-file character*. Note that even though `Ctrl-D` specifies an uppercase `D`, you can type a lowercase `d` here:

```
venus% ^D

venus login:
```

Turning off the workstation does not necessarily log you out. Most Sun systems do not have a ''time-out'' mechanism: unless you log out explicitly, you will probably remain logged in to the system.

After you log out, the system displays the login prompt again.

The following table may help if you have trouble logging out:

Table 2-1 *Problems Logging Out*

Problem	*Solution*
System says: `There are stopped jobs`	Type: `logout` two or three times
System says: `Not login shell`	Type: `exit` then type: `logout`
You're just completely stuck. (This might happen when logged into a machine over the network.)	Try typing: `Return` ~ . `Return`

3

Files: Basic Concepts

This chapter introduces some basic concepts having to do with *files*. Chapter 4 presents commands for manipulating files. If you're familiar with the following, you may want to skip to Chapter 4:

- Different kinds of files;
- Directories and filesystems;
- The SunOS file hierarchy and parent vs. child directories;
- Absolute vs. relative pathnames; and
- Your home directory.

3.1. What Is a File?

The *file* is the basic unit of the SunOS operating system. Almost everything is treated as a file, including:

- *Documents.* These include text files, such as letters, computer source code, or anything else you might write.
- *Commands.* Most commands are *executable* files; that is, they are files that you can run. For example, the `chess` command, which (as you might guess) plays chess, is an executable file.
- *Devices.* The SunOS operating system treats your terminal, your printer, and your disk drive as files.
- *Absolutely nothing.* There's even a special file, called `/dev/null`, that is a "black hole"—any sort of data sent there is discarded.
- *Directories.* A directory is a file that contains other files. Section 3.2 explains them further.

3.2. Files and Directories

As mentioned above, a directory is a file that holds other files. Directories can also contain other directories, that can contain other directories, and so on. You move around from directory to directory in order to manipulate the files they contain.

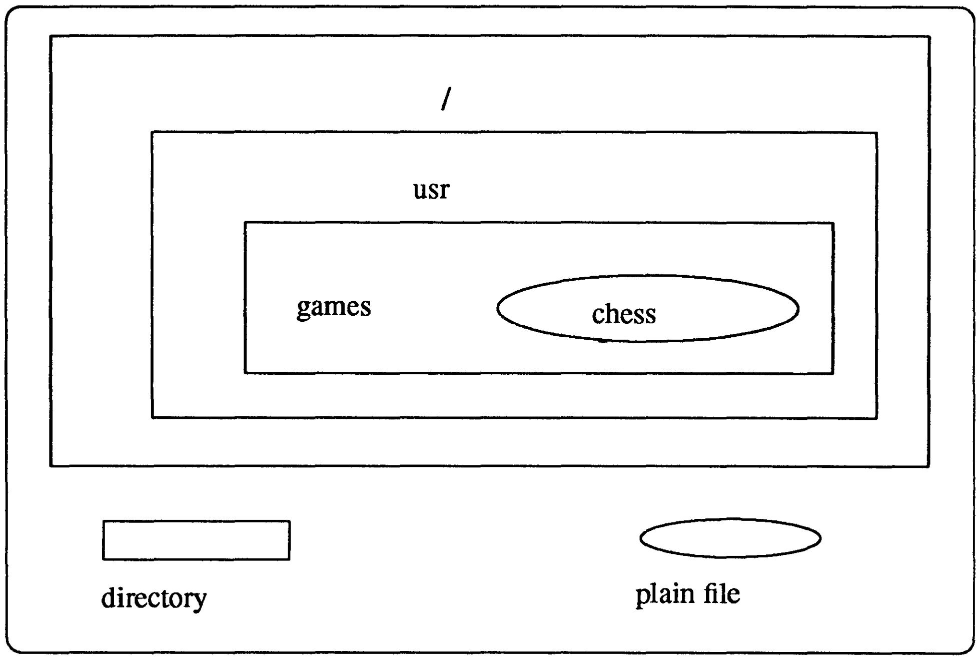
The symbol / has two meanings: it's used to separate a file's name from the name of the directory it's in, and it also is the name of the very first (or "top" or "`root`") directory in the operating system.

The `chess` game is located in a directory called `games`, which is itself located in a directory called `usr`, which is located in a directory called / (pronounced "slash"). We use the / symbol to separate the names of directories and files; here's how we represent the location of `chess`:

```
/usr/games/chess
```

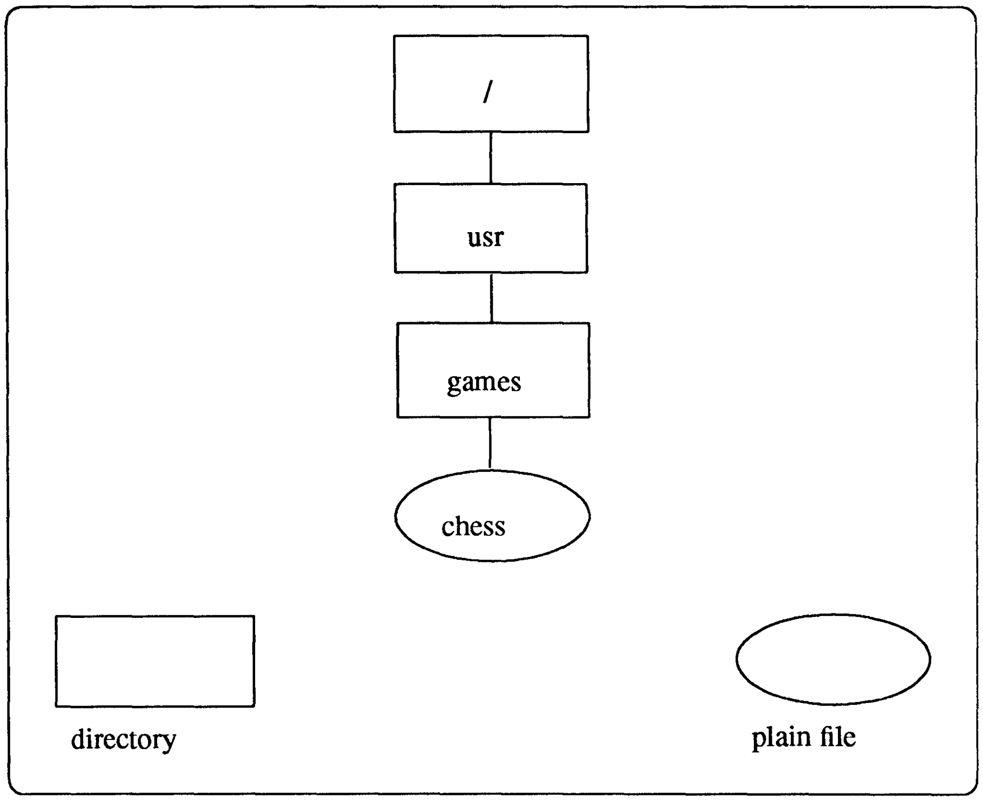
Here's one way to visualize the layout:

Figure 3-1 *Visualizing Files and Directories*



However, it's better to visualize the system as a *hierarchy*:

Figure 3-2 *A Simple File Hierarchy*



Directories: Parents and Children

Figure 3-2 bears a passing resemblance to a family tree. Indeed, we use the term *parent directory* to describe a directory that contains another directory. The directory it contains is a *child directory*, also called a subdirectory. In Figure 3-2, `usr` is the parent directory of `games`; likewise, `games` is the child directory of `usr`.

3.3. Pathnames: Absolute and Relative

The name of the chess program is `chess`. `/usr/games/chess` is the *pathname* of the chess program; that is, it's the name of the file (`chess`) plus its location (`/usr/games`). If `chess` were located in `/usr/lib` (a different directory), its pathname would be

```
/usr/lib/chess
```

To run the chess program, type:

To find out how to play `chess`, type **`man chess`**. Type Ctrl-D to quit the chess game.

```
venus% /usr/games/chess
(the chess games begins)
```

Pathnames are important because you can't use a file unless you can find it.

Now we come to a bit of a tricky concept.

/usr/games/chess is an *absolute pathname.* This means that it gives the location of chess in terms of its overall position in the whole SunOS file hierarchy. To find chess, you go up to /, and then back down through usr and games.

(We normally say that chess is located in the directory /usr/games.)

Relative Pathnames

It's inconvenient to have to give the full, absolute pathname every time you want to access a file. A *relative pathname* describes the file's location not in terms of the whole SunOS hierarchy, but *relative to your location in that hierarchy.*

Suppose you're in the directory /usr.[1] Since this directory contains the directory games, which contains the chess program, you don't have to specify the /usr part—you're already there. So instead of playing chess by typing /usr/games/chess, you can play by just typing:

```
venus% games/chess
(the chess game begins)
```

By the same token, if you're located in /usr/games, you need only type:

```
venus% chess
(the chess game begins)
```

You can't move to /usr/games/chess, because it's a file, not a directory.

Relative Pathname to Play Chess	
If You're Located In...	**...The Command Is:**
anywhere	/usr/games/chess (*absolute pathname*)
/	usr/games/chess
/usr	games/chess
/usr/games	chess

3.4. Filesystems

The SunOS filesystem is a hierarchical collection of a great number of directories and files. Some files are private to one machine, some files are shareable between different kinds of machines, and some files are shareable by only the same kind of machine.

Your home directory, and all the files that descend from it, are part of the larger filesystem. At the same time, your home directory and its files constitute a filesystem. In this sense, a filesystem can be considered any parent directory and its children.

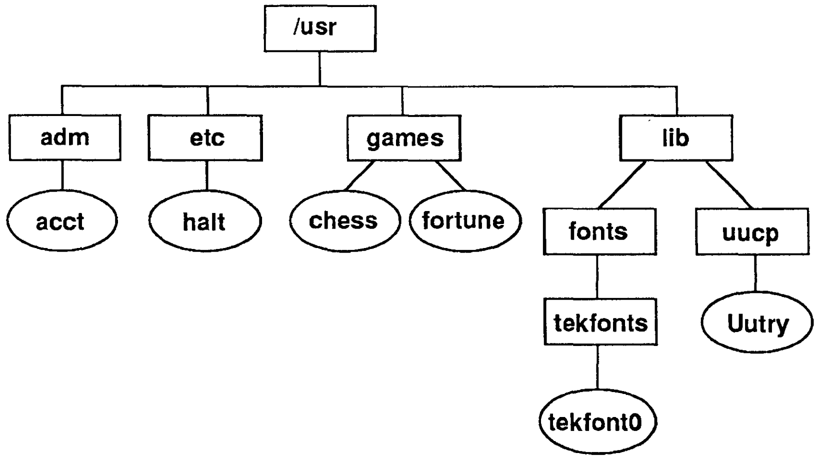
Figure 3-2 shows a very simple hierarchy. You'd need a very large piece of paper to show *all* the files and directories located under /.

Figure 3-3 shows a very stripped-down /usr filesystem. In point of fact, it is much, much larger, with many more files and directories, but for the purposes of

[1] Chapter 4 explains how to move around from directory to directory.

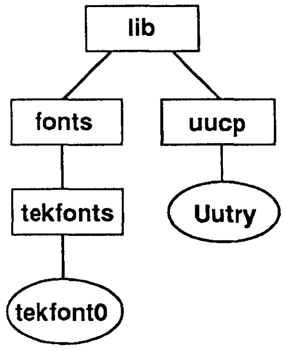
this discussion we will limit ourselves to just what's shown in the figure.

Figure 3-3 *The* `usr` *Filesystem (Partial)*



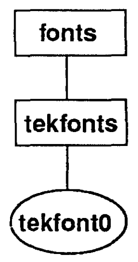
The `/usr` filesystem (as depicted here) contains other filesystems, as well. One such filesystem is `/usr/lib`. The `/usr/lib` filesystem contains the directories `fonts` and `uucp`, plus everything these two directories contain:

Figure 3-4 *The* `/usr/lib` *Filesystem (Partial)*



`/usr/lib` itself contains the `/usr/lib/fonts` filesystem. `/usr/lib/fonts` contains the directory `tekfonts`, which contains the file `tekfont0`.

Figure 3-5 *The* `/usr/lib/fonts` *Filesystem (Partial)*



3.5. Your Home Directory

You have a home directory that you go to when you log in. This is the starting point for your own filesystem, containing your own files.

To find out what your home directory is, type as follows:

```
venus% cd
venus% pwd
(your home directory is displayed)
```

You can create files and subdirectories in your home directory.

3.6. Useful Abbreviations

The SunOS operating system has some useful abbreviations. Their usefulness will become more apparent in Chapter 4, but you should take a moment to glance at them now. They'll come in handy.

.	The current directory, i.e., the directory you're in.
..	The directory "above" the one you're in (the parent directory). Useful when you want to move or operate near your current position in the file hierarchy.
~	Your home directory. Allows you to operate on files and directories relative to your home base.
?	Any single character in a filename.
*	Any group of characters in a filename.

3.7. The SunOS File Hierarchy

To find out about the various files and directories that make up the SunOS operating system, check the `hier` and `filesystem` pages in Section 7 of the *SunOS Reference Manual.* You can also get this information by typing at a system prompt

```
venus% man hier
```

or

```
venus% man filesystem
```

4

Manipulating Files

This chapter gives some useful commands for manipulating files. We assume you are already familiar with most of the concepts in Chapter 3. You may want to review that chapter before reading this one.

This chapter covers

- Changing directories;
- Finding your place in the file hierarchy;
- Listing the contents of a directory;
- Creating, removing, moving, and copying files and directories;
- Printing files;
- File permissions and ownerships; and
- Some useful abbreviations.

Here's an overview of the commands discussed in this chapter:

Table 4-1 *Basic Commands for Manipulating Files*

File Manipulation Commands	
Command	**Action**
`cd`	moves you to another directory
`pwd`	tells which directory you're in
`pushd`	moves to directory and creates stack
`popd`	returns through directory stack
`dirs`	displays directories in stack
`mkdir`	creates a directory
`rmdir`	removes a directory
`ls`	lists the contents of a directory
`file`	displays the type of file
`more`	displays the contents of a file
`cat`	displays the contents of a file
`cp`	copies a file
`rcp`	copies a file to & from other machines
`mv`	moves or renames a file
`rm`	removes a file
`touch`	creates or updates a file
`lpr`	prints a file
`lpq`	checks printer status
`lprm`	removes a printing job
`chmod`	sets permissions on a file
`umask`	sets default permissions
`chown`	changes ownership of a file
`ln`	makes links between files

Additionally, this chapter covers the following abbreviations:

Table 4-2 *Abbreviations for Files and Directories*

Useful Abbreviations	
`.`	the current directory
`..`	the "parent" directory
`~`	home directory
`?`	a single character in a file's name
`*`	a group of characters in a file's name

4.1. Moving Around: Changing Directories With `cd`

In order to manipulate files, you need to be able to get to them. The `cd` command moves you from directory to directory. Its syntax is

```
cd directory
```

where *directory* is the name of the directory to which you want to go.

Remember to press [Return] after typing commands.

Here's how you would go to the directory `/usr/games`:

```
venus% cd /usr/games
venus%
```

The cd command by itself (without a directory) moves you to your home directory:

```
venus% cd   (move to home directory)
venus% cd /usr/games    (move to /usr/games)
venus% cd   (back to home directory)
venus%
```

4.2. Where Am I Now? The pwd Command

The pwd (for print working directory) command tells you where you are when you change directories.

In the example below, assume that your home directory is /home/venus/medici:

```
venus% pwd
/home/venus/medici
venus% cd /usr/games
venus% pwd
/usr/games
venus% cd   (go to home directory)
venus% pwd
/home/venus/medici
venus%
```

4.3. Interlude: Two Abbreviations (~ and ..)

The SunOS operating system has a number of abbreviations that can make life a little simpler for you. Let's look at two.

The Home Directory (~)

Your home directory may be abbreviated as

~

The notation

~/*file*

refers to a file in your home directory. (This is also true for directories in your home directory.) The notation

~*user*

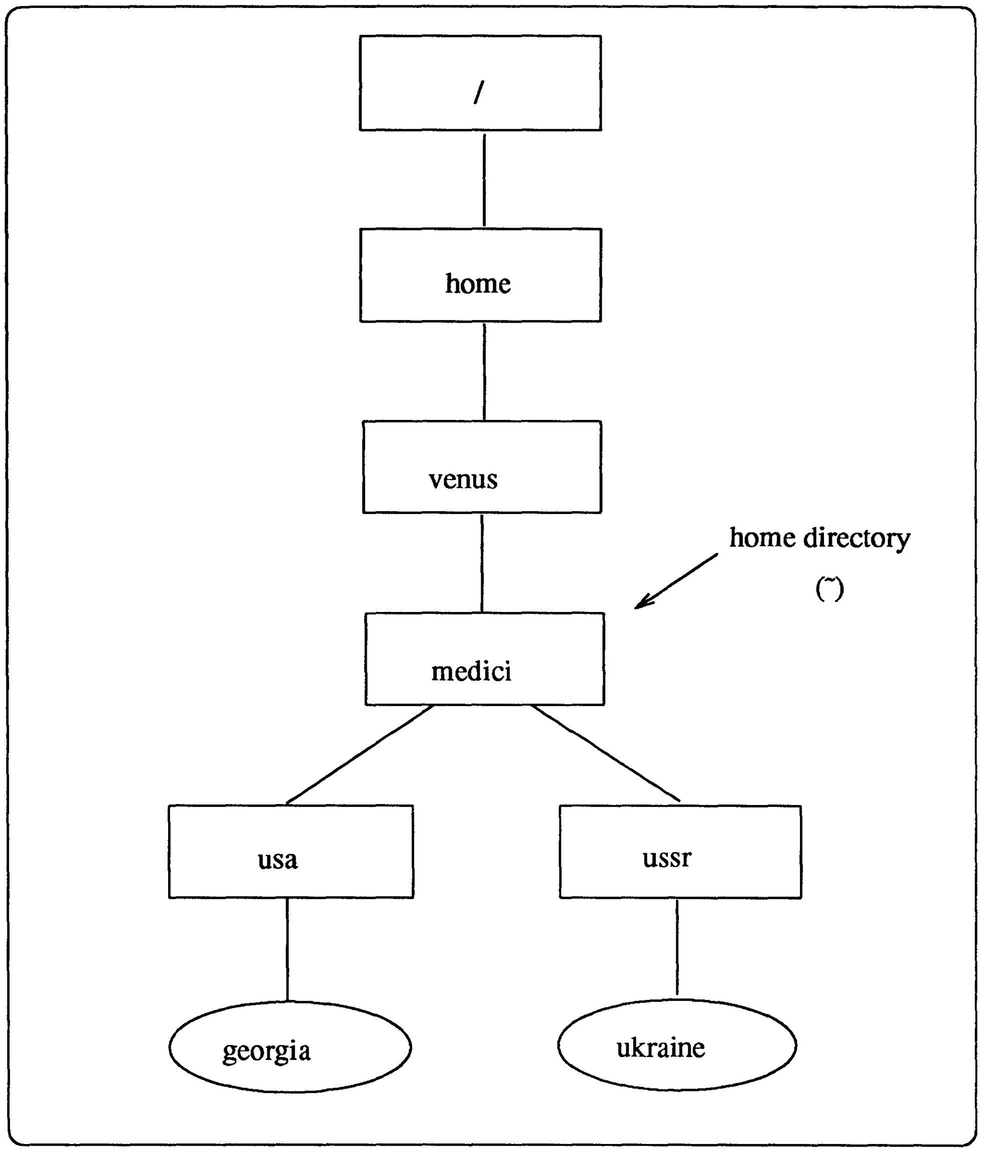
refers to the home directory of another user. Therefore, the notation

~*user*/*file*

refers to a file located in that user's home directory.

Suppose your home directory is /home/venus/medici and that it contains two subdirectories, called usa and ussr. The directory usa contains the file georgia, while the directory ussr contains the file ukraine. This is illustrated in Figure 4-1:

Figure 4-1 *A Home Directory*



If you want to get to `ussr`, you could type

```
venus% cd /home/venus/medici/ussr
venus%
```

but it's easier to type

```
venus% cd ~/ussr
venus%
```

instead.

The Parent Directory (..)

The abbreviation .. refers to the directory just above the one you're in (known as the *parent directory*).

Suppose that you've moved to /home/venus/medici/ussr. If you want to move up one directory, simply type cd .., as shown below:

```
venus% pwd      (where am I?)
/home/venus/medici/ussr
venus% cd ..
venus% pwd      (where am I now?)
/home/venus/medici
venus%
```

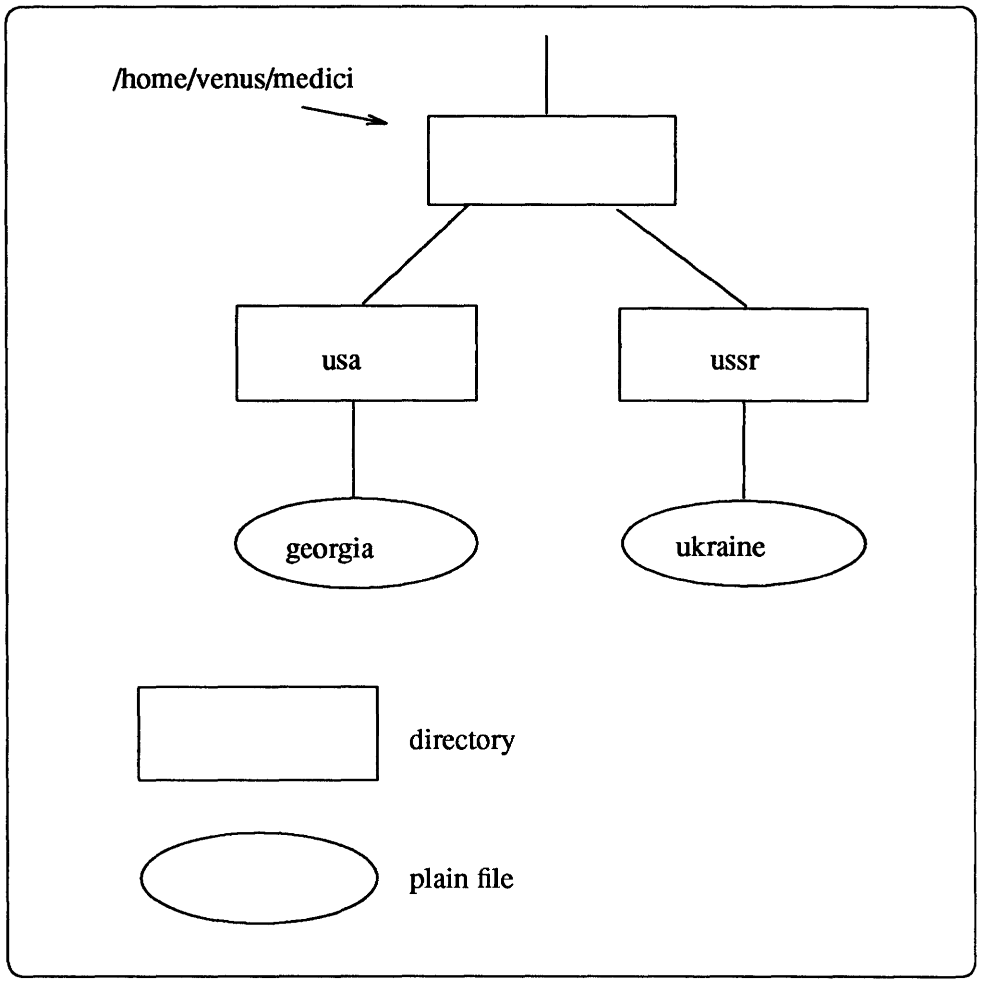
You're now in /home/venus/medici.

See Section 3.3 for more on relative *vs.* absolute pathnames.

The abbreviation .. is a *relative pathname*. The directory above the one your in is called its *parent directory*.

Suppose once again that you're in /home/venus/medici/ussr. This time you want to move over one step laterally (not vertically) to usa.

Figure 4-2 *Moving Laterally With* `..`



One way to move from `ussr` to `usa` is to type

```
venus% cd /home/venus/medici/usa
venus%
```

That's using an *absolute pathname*. An easier way is to use a relative pathname. Since you're already in `ussr`, you can type

```
venus% pwd     (where am I?)
/home/venus/medici/ussr
venus% cd ../usa     (move sideways)
venus% pwd     (where am I now?)
/home/venus/medici/usa
venus%
```

"`cd ../usa`" means "go up one directory, and then back down to `usa`."

4.4. More Shortcuts to Changing Directories: `pushd`, `popd`, and `dirs`

Using the `cd` command to change directories has a disadvantage when you're switching directories often: it doesn't remember where you've been. As an alternative to changing directories with `cd`, you can create a directory stack — an ordered listing of directories you've used — and then return to these directories without having to recall complicated pathnames.

These commands are available only in the C shell; see Chapter 7 and *SunOS User's Guide: Doing More.*

Three related commands let you do this:

- `pushd`: when you're in a directory that you know you'll want to return to, use this command to *change* to a new directory and *add* the current directory to the directory stack. Use the command form

  ```
  pushd directory
  ```

  where *directory* is the directory you want to move to.
- `popd`: use this command to work your way back through the directory stack, a directory at a time. Note that this command effectively dismantles the stack you created by changing directories with `pushd`.
- `dirs`: use this command to see a list of the directories in the stack. `dirs -l` display the full pathnames of stacked directories. Note that `pushd` and `popd` also display the directory stack, with the current directory at the left.

The example below shows all three commands in use:

```
venus% pwd (checks name of current directory)
/var/spool/mail
venus% pushd ~
~ /var/spool/mail
venus% pushd /etc
/etc ~ /var/spool/mail
venus% dirs
/etc ~ /var/spool/mail
venus% popd
~ /var/spool/mail
venus% popd
/var/spool/mail
```

4.5. Making New Directories With `mkdir`

The `mkdir` command creates directories. Its syntax is

```
mkdir directory
```

where *directory* is the name of the directory you want to create.

Suppose your home directory is `/home/venus/medici` and you want to create a directory there called `letters` to store copies of the letters you write.

```
venus% cd   (go to home directory)
venus% mkdir letters
venus%
```

You could have used `mkdir` without changing directories. In this case, you must give the full, or absolute, pathname of the directory you want to create. This is true of all file-manipulating commands.

```
venus% mkdir /home/venus/medici/letters
venus%
```

See Chapter 3 for more on relative *vs.* absolute pathnames. In this case, the shortest way to create `letters` is to use the ~ abbreviation, explained in Section 4.3.

```
venus% mkdir ~/letters
venus%
```

The commands in the three examples above are equivalent.

Sometimes you'll be denied permission when you try to create a directory. One of the most common problems is when you see the message:

```
mkdir: Permission denied.
```

or something similar. Often the problem is that the directory containing the one you want to create doesn't have the right permissions. See Section 4.15 for information on permissions.

For more help with problems, see *SunOS User's Guide: Basic Troubleshooting*.

4.6. Removing Directories

`rmdir` does the opposite of `mkdir`: it removes existing directories. Its syntax is

```
rmdir directory
```

where *directory* is the name of the directory you want to remove.

```
venus% rmdir ~/letters
venus%
```

A directory *must be empty* before you can remove it. It must contain no files and no directories that contain files. (Section 4.12 explains how to remove files.)

You can remove a directory and all its contents, including subdirectories and all *their* contents, with the `rm -r` command.

CAUTION **Be very careful when using this command! You may accidentally wipe out whole hierarchies of important files.**

```
venus% rm -r ~/letters
venus%
```

There's a safer way to remove a directory and its contents. Use `rm -ir` to have the operating system query you for confirmation before it removes any files:

```
venus% rm -ir ~/letters
remove ~/letters?  y
remove ~/letters/subdirectory/otherfile?  n
venus%
```

4.7. What Files Are in This Directory? (`ls`)

The `ls` command tells you what files (and directories) are in a directory. Next to the `cd` command, `ls` is probably the most-used command in the SunOS operating system.

The syntax `ls` is

```
ls directory
```

where *directory* is the name of the directory you want to look at.

```
venus% ls /var
adm                log                spool              yp
crash              preserve           tmp
venus%
```

If you don't specify a directory, `ls` lists the contents of the directory you're in.

```
venus% ls
alabama            colorado           idaho              iowa
alaska             georgia            illinois
california         hawaii             indiana
venus%
```

Seeing Hidden Files (`ls -a`)

Files starting with a dot (`.`) are *hidden*. That means that `ls` won't normally display them. However, `ls -a` does (the `a` stands for "all").

.. represents the parent directory;
. is the current directory.

```
venus% ls
artichokes         bananas            cabbage
venus% ls -a
.                  .chsrc             artichokes         cabbage
..                 .rootmenu          bananas
venus%
```

Why would you want to hide a file? Generally, because the file is important enough that you want to keep it around—it may get used a lot—but there's no reason to have `ls` display it under most circumstances.

(One example of a hidden file is .cshrc, kept in your home directory. It contains parameters governing how the SunOS operating system behaves for you. See *SunOS User's Guide: Customizing Your Environment* for more on .cshrc.)

What Kinds of Files Are These? (ls -F)

ls followed by -F can also tell you what kind of files are in a directory. There are five file types displayed by ls -F:

Directories
: ls appends a slash (/) to the filename to indicate that this file is really a directory.

Executables
: ls appends an asterisk (*) to the filename to indicate that this file is *executable*; that is, it's a program you can run.

Plain Files
: If only the file name appears, with nothing appended, then the file is a plain file.

Links
: ls appends an "at" symbol (@) to indicate a link. Links are explained in Section 4.16.

AF_UNIX Sockets
: Indicated by a trailing equals sign (=). These are for specialized programming applications.

```
venus% ls -F
animals/        go_carts*       letter.1        states/
food/           hangman*        letter.2
venus%
```

The example above shows that the directory contains three subdirectories (animals, food, and states), two programs (go_carts and hangman), and two plain files (letter.1 and letter.2).

What Kinds of Files Are These? (file)

The file command gives you more information than ls -F about files:

```
venus% file go_carts animals letter.1
go_carts:   sparc demand paged dynamically linked executable
animals:    directory
letter.1:   English text
```

Note that, like most SunOS commands, you can use file on more than one file at a time.

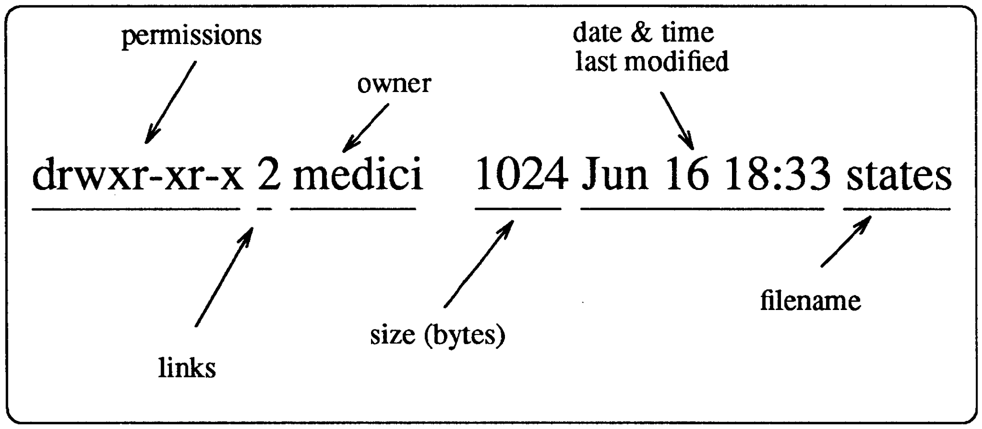
Full Information on Files (`ls -l`)

`ls -l` (lowercase "L") yields yet more information about the contents of a directory, including who owns a file, its size, when it was last modified, and so on.

```
venus% ls -l
total 108
-rwxr-xr-x  1 medici     133490 Jun 16 18:30 go_carts
-rw-r--r--  1 medici       8111 Jun 16 18:31 letter.1
drwxr-xr-x  2 medici       1024 Jun 16 18:33 states
venus%
```

Here's how we break down this information:

Figure 4-3 *Information Provided by* `ls -l`


permissions
date & time last modified
owner
drwxr-xr-x 2 medici 1024 Jun 16 18:33 states
links
size (bytes)
filename

And here's what those things mean:

Permissions
What you can do with this file. See Section 4.15 for more on permissions. For now, just note that if the file is a directory, the first letter here is a `d`.

Links
Links are discussed in Section 4.16; basically, think of each file (or filename, including directories) as taking up a certain number of allotted spots on the system.

Owner
The owner of the file. See Section 4.15 for more on ownership. For now, just note that in most cases the owner is the person who created the file.

Size
The size of the file in bytes.

Time of modification
When the file was created, edited, or otherwise changed.

Name
The name of the file or directory.

Information on Directories (`ls -ld`)

Typing `ls -ld` gives you the same information as `ls -l`, except that it does so for the directory itself, not its contents.

```
venus% ls -ld /usr/lib
drwxr-sr-x 21 bin         3072 Dec 15  1988 /usr/lib
venus%
```

Combining Options

You can combine all the options to `ls` that we've covered. (There are many more options to `ls`; see the *SunOS Reference Manual* or type `man ls` at a system prompt.) The syntax is

`ls -`*`options`* *`directory`*

where *options* are one or more of `l`, `a`, or `F`, as discussed above, and *directory* is the name of a directory to look at. (If you don't specify a directory, you look at the one you're in.)

```
venus% ls -laF
drwxr-xr-x  5 medici        512 Jun 19 10:23 ./
drwxr-xr-x  5 medici        512 Jun 16 18:30 ../
-rw-r--r--  1 medici       1234 Jun 19 10:23 .cshrc
-rw-r--r--  1 medici      12272 Jun 19 10:23 .rootmenu
drwxr-xr-x  2 medici        512 Jun 16 18:32 animals/
drwxr-xr-x  2 medici        512 Jun 16 18:32 food/
-rwxr-xr-x  1 medici     133490 Jun 16 18:30 go_carts*
-rwxr-xr-x  1 medici      23441 Jun 16 18:30 hangman*
-rw-r--r--  1 medici       8111 Jun 16 18:31 letter.1
-rw-r--r--  1 medici       2130 Jun 16 18:31 letter.2
drwxr-xr-x  2 medici       1024 Jun 16 18:33 states/
```

In the example above, the options `-l`, -a, and `-F` are combined: `-l` causes `ls` to print out full information on the files; `-F` displays the type of file after the filename; and `-a` displays hidden files (ones starting with a dot).

Table 4-3 `ls` *and Its Options*

Command	Action
`ls`	shows files in a directory
`ls -a`	shows hidden files
`ls -F`	shows file types
`ls -l`	shows full information on files
`ls -ld`	shows full information on directories

4.8. Interlude: Two More Abbreviations (`*` and `?`)

The SunOS operating system allows you to work with more than one file at once by providing two *wild card characters*, so called because they stand for any character, or group of characters, in a file's name. In this way, you can operate on a number of files whose names match the pattern you give.

The * Wild Card

An asterisk by itself represents all the files in a directory, *except hidden ones.*

An asterisk (*) represents any number of characters in a file's name.

Suppose you have a directory, states that contains files with names of states. The following command would give you a list of all files starting with m:

```
venus% ls states/m*
maine          michigan       mississippi    montana
maryland       minnesota      missouri
venus%
```

The next command would give a list of files starting with m and ending with a:

```
venus% ls states/m*a
minnesota      montana
venus%
```

The ? Wild Card

The question mark (?) wild card represents any single character in a file's name. Therefore, you can use it to select a group of files matching a given pattern, with the same number of letters in them.

This command would list all files with names seven characters long, and that begin and end with a:

```
venus% ls states/a?????a
arizona      alabama
venus%
```

Note that ls didn't display alaska, because the word has too few characters to match the pattern.

4.9. Viewing Files With more

The more command displays the contents of a file:

```
venus% more filename
```

more displays one screenful at a time; if the file is more than one screen long, it displays the word

```
—More—
```

followed by the percentage of the file displayed so far, at the bottom of the screen.

Press the [Space Bar] to see the next screen of the file. Press q to quit.

To search for a string, type /, and then the string to be searched for.

Press h for help (this displays a list of commands). Type **man page** at a prompt for additional information.

The cat Command

The cat command also displays the contents of a file, but not page-by-page. Chapter 7 shows some ways in which cat is more useful than more.

4.10. Copying Files and Directories

Copying files and copying directories is pretty much the same, whether you're copying on the same machine or from one machine to the next. We introduce another abbreviation (`.`) that is especially useful here. Note that *copying* a file isn't the same as *moving* a file. When you copy a file, you leave a copy of it where it was.

Copying Files (Same Machine)

To copy files on the same machine, use the `cp` command. `cp` has the following syntax:

```
cp source-file  destination-file
```

where *source-file* is the name of the file you want to copy, and *destination-file* is the name of the file you want it copied to.

Here's how you make a copy—with a new name— of a file in the same directory:

```
venus% cp georgia georgia.copy
venus%
```

If you're copying a file to another directory—without changing its name—the syntax is:

```
cp source-file  directory
```

where *directory* is the place to which you're copying the file.

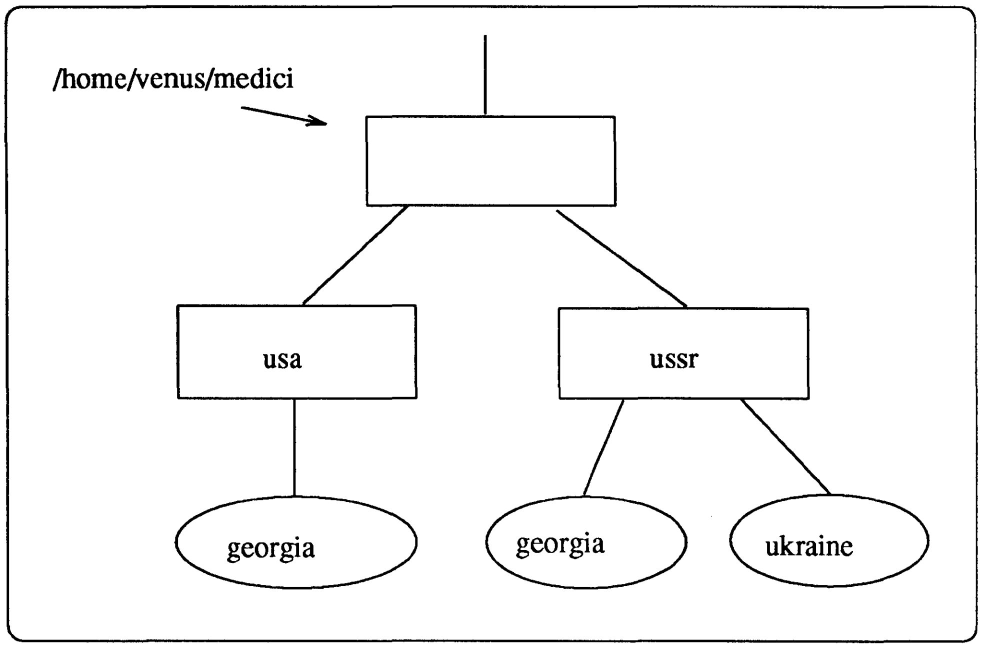
Here's how you copy a file—keeping the same name—to another directory. Suppose you have the filesystem shown in Figure 4-2:

~ represents your home directory.

```
venus% cp ~/usa/georgia ~/ussr
venus%
```

This is the result (again, compare this with Figure 4-2):

Figure 4-4 *Copying a File to Another Directory (Same Name)*



And here's how you make a copy—with a *new* name—in another directory (in this case, /tmp):

```
venus% cp ~/states/georgia /tmp/georgia.copy
venus%
```

To copy the contents of one directory to another, use the * abbreviation explained in Section 4.8. This is the way you would copy all the files starting with m and ending with a, in the directory states, into the directory /tmp:

```
venus% cp ~/states/m*a /tmp
venus% ls -l /tmp/m*a     (what was copied?)
minnesota        montana
venus%
```

Copying Directories (Same Machine)

Use cp -r to copy an entire directory, plus all its subdirectories, into another directory. The syntax is the same as above:

```
cp -r source destination
```

where *source* is the directory you're copying and *destination* is the directory you're copying to. This is how you'd copy your entire home directory and all its subdirectories and *their* contents into the directory /tmp:

```
venus% cp -r ~ /tmp
venus%
```

Interlude: The . Abbreviation

A single dot (.) represents whatever directory you're currently in (called the current or working directory)> This abbreviation comes in handy when copying files. For example, if you want to copy the file `chess` from `/usr/games` to the directory you're in, you type:

```
venus% cp /usr/games/chess .     (note final dot)
venus%
```

Copying Files (Different Machines)

To Your Machine

Use `rcp` to copy from one machine to another. `rcp`'s syntax is similar to `cp`'s.

To copy from another machine to yours, the syntax is:

`rcp` *`machine:source destination`*

where *machine* is the name of the other machine; *source* is the name of the file(s) you want to copy; and *destination* is the place on your machine to which you're copying.

The example below shows how to copy the file `/home/earth/terran/notes` from the machine `earth` to the directory `/tmp`:

```
venus% rcp earth:/home/earth/terran/notes /tmp
venus%
```

The example below is similar to the previous one. We include it here so that you can see how to combine various abbreviations and syntaxes. It shows how you copy all the files ending in `.c` from user `holly`'s home directory on the machine `mars` to your current directory on your machine:

`~holly` is Holly's home directory; `*.c` represents files ending in *.c*, and `.` represents your current directory.

```
venus% rcp mars:~holly/*.c .
venus%
```

From Your Machine

To copy files from your machine to another, simply reverse the syntax:

`rcp` *`file(s) machine:destination`*

where *file(s)* are the file(s) you want to copy; *machine* is the name of the other machine; and *destination* is the place (directory) on the other machine where the file's to go.

The example below shows how you'd copy the file `buzzcut` from your directory `~/haircuts/usa` to the directory `~holly/trivia` on the machine mars:

~ is your home directory; ~holly is user holly's home directory .

```
venus% rcp ~/haircuts/usa/buzzcut mars:~holly/trivia
venus%
```

Copying Directories (Different Machines)

Use `rcp -r` to copy subdirectories and their contents.

4.11. Moving (Renaming) Files

Use the `mv` command to move files.

Moving files is similar to copying files, except that it removes the old version of the file. And renaming a file is the same as moving it; you are simply "moving" it to a new name.

The syntax for `mv` is

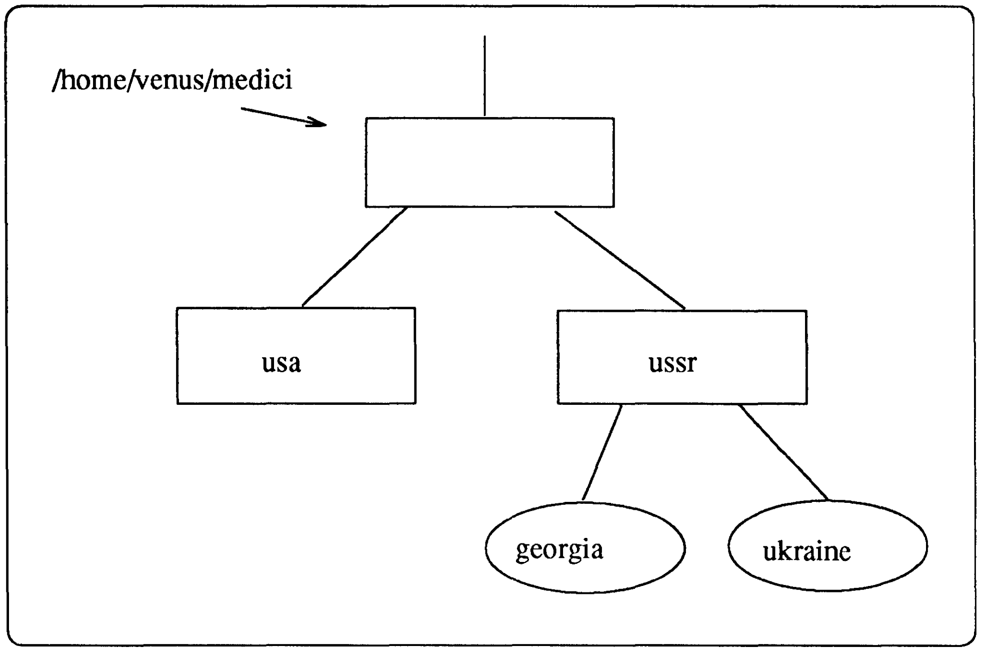
```
mv old new
```

where *old* is the old filename and *new* is the new location or name.

The example below shows how you move a file (`georgia`) from one directory to another (`usa` to `ussr`).

```
venus% mv ~/usa/georgia ~/ussr
venus%
```

Figure 4-5 *Result of Moving a File*



Renaming Files

Technically, we renamed the file ~/usa/georgia as ~/ussr/georgia. Usually, though, we think of this as moving it from usa to ussr. That's why we use the mv command to rename files. This is how you rename the file agatha as bernice:

```
venus% mv agatha bernice
venus%
```

4.12. Removing Files

Use rm to remove a file or files. Here's how you remove all files starting with the letter m and ending with the letter a:

```
venus% rm m*a
venus%
```

CAUTION **Be very careful using this command! Once you remove a file, you can't get it back.**

A safer way to remove files is to add the -i option to rm. This causes the operating system to query you for confirmation before deleting a file:

```
venus% rm -i m*a
rm: remove minnesota?  y
rm: remove montana?  n
venus%
```

4.13. Creating Files

There are a number of ways to create files.

- *Copying or moving an existing file.* (You've already learned how to do this.)
- *Using an editor.* This is the most common way to make a file; it's how you compose letters, write programs, create documents. Chapter 5 explains vi, the basic SunOS editor.
- *Using the output of a program.* The cc command, for example, produces executable ("runnable") programs from the C programming language.
- *Using the* touch *command.* touch creates empty files, or updates ("time-stamps") existing ones.
- *Using the* cat *command.* When you want to create a text file but don't want to use an editor—say, the file is very small and you want to do it fast—use the cat command.

  The syntax is

  ```
  cat > filename
  ```

  where *filename* is the name of the file you want to create.

  First, type the command. Then enter the text you want, pressing [Return] at the end of each line. End your input by pressing [Ctrl-D] on a new line.

```
venus% cat > dogs
Of all breeds, the rare German Schnaubel-
Gotterdammerung is most prized by collectors.
It has wiry hair, short legs, and pince-nez
glasses.  It can also sing Cole Porter, although,
to be fair, only when it's "had a few."
Ctrl-D
venus%
```

4.14. Printing Files

This section introduces a few of the commands used in printing.

Table 4-4 *Printing Commands*

Printing Commands	
Command	**Action**
`lpr`	prints out unformatted or `nroff`-formatted file
`lpr -t`	prints `troff`-formatted file
`lpr -P`*`printer`*	prints out to named *printer*
`lpr -h`	prints out without a header page
`lpq`	reports on the status of the printer
`lprm`	removes a printing job

Printing With `lpr`

Use `lpr` to print out files. This command will send unformatted files to your default printer.

```
venus% lpr textfile
venus%
```

Sending to a Different Printer

To send a job to a different printer, use `lpr -P`. The syntax is

`lpr -P`*`printer filename(s)`*

where *printer* is the name of the other printer and *filename* is the name of the file(s) to print.

```
venus% lpr -Pprinter2 textfile
venus%
```

Printing Formatted Text

Use `lpr` to print out files formatted with `nroff`. Use `lpr -t` to print out text you've formatted with `troff`. (Chapter 6 introduces `nroff` and `troff` and discusses the use of macro packages in formatting and printing text.)

```
venus% troff t_source > t_output     (create formatted file)
venus% lpr -t t_output    (print it)
venus%
```

Printing Without Header Pages

Normally when you print a file, you also produce a cover sheet that identifies the output. This is useful if your file is going to get printed along with a lot of others or if you don't want people reading what you're printing. Such sheets are called *header pages*.

When it's *not* important to have a header page, you can save part of a tree by using `lpr -h` to specify that you don't want one printed.

```
venus% lpr -h textfile
venus%
```

Checking the Printer Status

When you print a file, you send a copy of the file to a *print queue*. A print queue is all the files waiting to be printed on that printer.

Each file waiting to be printed is called a *printer job*. Each printer job has a number associated with it.

The `lpq` command tells you what's in the print queue and whether the printer is printing correctly or not.

You can use the `-P`*printer* option to check the status of other printers.

```
venus% lpq
printer1 is ready and printing
Rank   Owner      Job  Files            Total Size
active medici     16   textfile         149576 bytes
venus% lpq -Pprinter2
printer2 is ready and printing
Rank   Owner      Job  Files            Total Size
active medici     16   textfile2        139879 bytes
active carlos     17   big_plans         34311 bytes
active scotty     18   proposal         102296 bytes
venus%
```

Removing Print Jobs

If you send something to the print queue and then decide not to print it, use the `lprm` command.

There are two ways to use `lprm`. One way is to remove a single print job at a time. First, use `lpq` to determine the job number associated with that file. Then type `lprm` followed by that job number:

```
venus% lpr textfile   (print file)
venus% lpq    (check printer)
lw is ready and printing
Rank   Owner      Job  Files              Total Size
active medici     16   textfile           149576 bytes
venus% lprm 16   (remove print job)
printer1: dfA016venus dequeued
printer1: cfA016venus dequeued
venus%
```

Another way is to remove all your print jobs at once. To do this, type `lprm` followed by a hyphen (-):

Of course, you may just want to put more paper in the printer. . . .

```
venus% lpr textfile letter    (print files)
venus% lpq    (check printer)
Printer Error: may need attention! (out of paper)
Rank   Owner      Job  Files            Total Size
active medici     16   textfile         139879 bytes
active medici     17   letter             4201 bytes
venus% lprm -   (remove all print jobs)
printer1: dfA016venus dequeued
printer1: cfA016venus dequeued
printer1: dfA017venus dequeued
printer1: cfA017venus dequeued
venus%
```

As with `lpr` and `lpq`, you can use the `-P` option with `lprm` to remove files from a specified printer.

4.15. Permissions and Ownership

Section 4.7 discusses the output of the `ls -l` command. Two things this output displays are *permissions* and *ownership*. These two concepts are very important to using the SunOS operating system.

Permissions

Permissions determine what users may do with a file or a directory. The following are the permissions associated with files and directories:

Readable

A file must be readable to be looked at or copied.

A directory must be readable for you to list its contents.

Writable

A file must be writable in order for you to modify it, remove it, or rename it.

A directory must be writable in order for you to add or delete files in it.

Executable (files only)

A file with executable permissions is one you can run, such as a program or a shell script.[2]

[2] You can give any file executable permission, but if the file isn't runnable as a program in the first place—

Searchable (directories only)

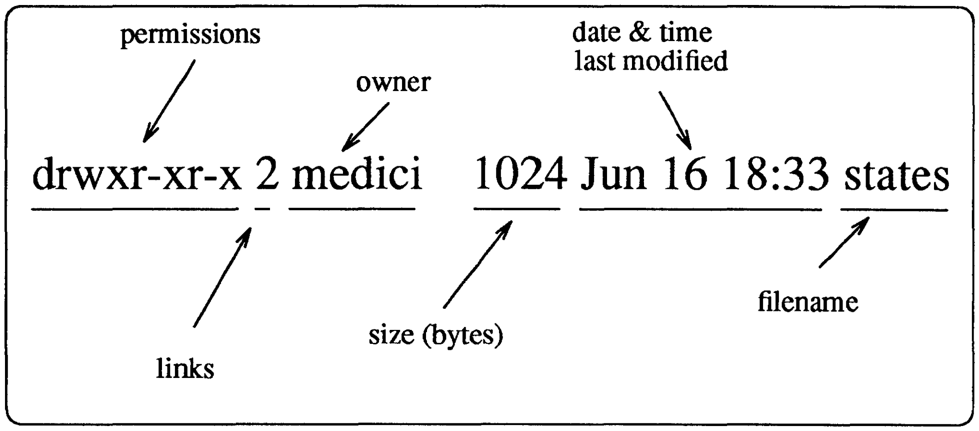
A directory must be searchable in order for you to `cd` to it, list its contents, or create or delete files there.

Determining Permissions

Use the `ls -l` command, as described in Section 4.7, to determine what permissions files and directories have.

Here's a close-up of part of the typical output of `ls -l`: note the area to the left, labeled "permissions." It is this region that tells you what permissions a file has.

Figure 4-6 *Information Provided by* `ls -l`


permissions
date & time last modified
owner
drwxr-xr-x 2 medici 1024 Jun 16 18:33 states
filename
size (bytes)
links

Now let's zoom in. Here's how the permissions area breaks down:

Figure 4-7 *Permissions Information*

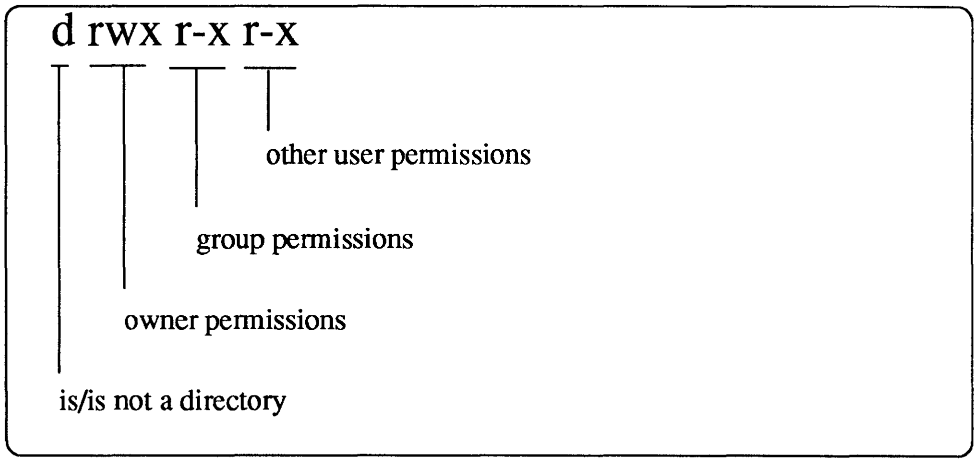


Figure 4-7, from left to right:

Is/is not a directory

If the letter `d` appears here, then this is a directory. If a hyphen appears

like an ordinary text file—it doesn't make sense for it to have executable permission. On the other hand, if you have a program that doesn't have executable permission, it won't run.

instead, then it's a file.

Owner permissions
These three letters tell what permissions the owner of the file has with respect to the file or directory. That is, whether he or she can read it, write it, execute it (file), or search it (directory).

Group permissions
These three letters tell what permissions users belonging to the same group as the file's owner have with respect to the file or directory. Groups are explained below.

Other user permissions
These three letters tell what permissions any other user has with respect to the file or directory.

What Those Letters Mean

Here's what each letter means.

`r` The file or directory is readable.

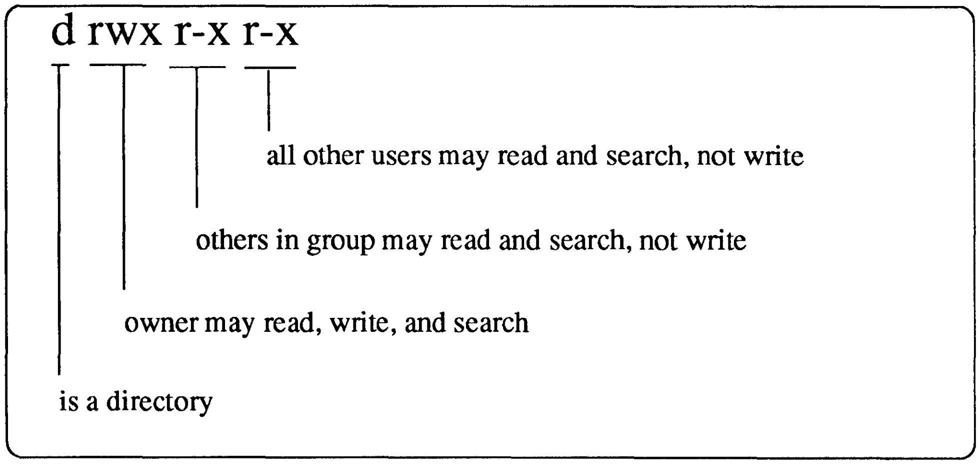
`w` The file or directory is writable.

`x` The file is executable, or the directory is searchable.

`-` The hyphen appears when the permission is switched off—if it appears in place of an `r`, then the file or directory is not readable; if it appears in place of a `w`, then the file or directory is not writable; and if it appears in place of an `x`, then the file isn't executable (or the directory isn't searchable).

This then is what Figure 4-7 means:

Figure 4-8 *Permissions Information*


d rwx r-x r-x
all other users may read and search, not write
others in group may read and search, not write
owner may read, write, and search
is a directory

Most files are created so that they are readable and writable by their owner, and readable by others.

Table 4-5 *Examples of Permissions Shown by* `ls -l`

Example	Meaning
`-rw-r--r--`	file is read/write for owner, read only for others
`-rwx------`	file is read/write/execute for owner only
`dr-xr-x---`	directory is read/search for owner and his group
`-rwxr-xr-x`	file is read/write/executable for owner, read/execute for others
`drwxr-x--x`	directory is read/write/search for owner, read/search for group, searchable only for others

Setting Permissions With `chmod`

Use `chmod` (**change mode**) to change the permissions on a file or directory. `chmod`'s syntax is

```
chmod who op permission filename
```

where:

Who
is the user(s) to change permissions for. The letter `u` means set permissions for the file's owner; `g` means to do so for the owner's group; `o` means other users not in that group; and `a` means all users.

Op is the operation to perform on the file or directory. `op` can be +, meaning add this permission for this person(s), or -, to take it away.

Permission
is `r` for readable, `w` for writable, and `x` for executable (files) or searchable (directories).

Filename
The name of the file or directory you're setting permissions on.

In the example below, we make the file `go_carts` readable and writable for all users (it already was readable):

```
venus% ls -l go_carts   (get current permissions)
-rw-r--r--  1 medici    1024 Jun 27 17:06 go_carts
venus% chmod a+rw go_carts
venus% ls -l go_carts
-rw-rw-rw-  1 medici    1024 Jun 27 17:06 go_carts
venus%
```

Now, we set `go_carts` to be executable for the its owner (`medici`):

```
venus% chmod u+x go_carts
venus% ls -l go_carts
-rwxrw-rw- 1 medici  1024 Jun 27 17:06 go_carts
venus%
```

And, finally, we make `go_carts` unreadable and unwritable for other users:

```
venus% chmod go-rw go_carts
venus% ls -l go_carts
-rwx------ 1 medici  1024 Jun 27 17:06 go_carts
venus%
```

Setting Default Permissions With `umask`

When you create a new file or directory, the system automatically assigns permissions. The default setting for new files is

```
-rw-r--r--
```

For new directories, the default is

```
drwxr-xr-x
```

You can change the default permission setting for the current session with the `umask` command:

`umask` *ogp*

where *o*, *g* and *p* are digits corresponding to the permission masks of the owner, group, and public, respectively.

You can change the permissions for all sessions by placing a `umask` command in your `.cshrc` file. See *SunOS User's Guide: Customizing Your Environment.*

`umask` uses three digits to determine the permissions. Permissions are computed according to the following table:

Table 4-6 *Values and Permissions for New Files*

Files		*Directories*	
value	Permissions	value	Permissions
0	`rw-`	0	`rwx`
1	`rw-`	1	`rw-`
2	`r--`	2	`r-x`
3	`r--`	3	`r--`
4	`-w-`	4	`-wx`
5	`-w-`	5	`-w-`
6	`---`	6	`--x`
7	`---`	7	`---`

`umask` does not activate *execute* permission for files.

Thus the command

```
umask 2
```

or

```
umask 002
```

yields permissions of `-rw-rw-r--` for files, and `drwxrwxr-x` for directories.

The command

```
umask 22
```

yields permissions of `-rw-r--r--` for files and `drwxr-xr-x` for directories.

Ownerships

Files generally belong to the person or process that creates them.

Changing File Ownership With `chown`

The ownership of a file can be changed with the `chown` command. Its syntax is

```
chown new-owner filename
```

NOTE *For security reasons, only the "superuser" can change ownership with* `chown`*. See SunOS User's Guide: Doing More.*

Here's how you can reassign the file `minnesota` to the user `holly`:

```
venus% ls -l minnesota
-rw-r--r-- 1 medici  30458 Jun 27 17:35 minnesota
venus% su
Password: (enter superuser password)
venus#
venus# chown holly minnesota
venus# ^D
venus% ls -l minnesota
-rw-r--r-- 1 holly   30458 Jun 27 17:35 minnesota
venus%
```

Groups

Each user may belong to one or more *groups*. Examples of groups are: everyone in marketing, all software engineers, or all users on a system. To find out which groups you belong to, use the `groups` command:

```
venus% groups
staff  doc
venus%
```

The `ls -lg` command is very similar to `ls -l`; in addition to showing you who owns a file, it tells what the group ownership of a file is:

```
venus% ls -lg
total 108
-rwxr-xr-x  1 medici  staff   1024 Jun 27 17:06 go_carts
-rw-r--r--  1 medici  staff     8111 Jun 16 18:31 letter.1
drwxr-xr-x  2 medici  staff     1024 Jun 16 18:33 states
venus%
```

4.16. Making Links

A *link* is a name associated with a file. The SunOS operating system allows several links to a file at any one time, so the same file can have more than one name. This is useful when you want to get at a file quickly from within different directories. Moreover, you can keep a link to a file in a restricted directory, thus allowing people access to the file without giving them access to the forbidden directory. When you create a file, the system makes the first link, or filename, for you. To make an additional link, use the `ln` command:

```
ln oldname newname
```

If you attempt to make a link to a file in a directory that is on a different disk or disk partition than that of *oldname*, you will get an error message of the form:

```
newname: Cross-device link
```

In this case, you can use the `-s` option of `ln` to make a *symbolic* link to the file:

```
ln -s oldname newname
```

A symbolic link is an entry in the directory that points to the *name* of another file, rather than to the file itself. A symbolic link can be made across devices, and can be made even when *oldname* does not exist. Because a symbolic link refers to another file's name, rather than to the file itself, it may be to your advantage to use a symbolic link instead of a regular link when you want to specify an alternate pathname to the same file.

Both regular (hard) and symbolic links allow you to use *newname* instead of *oldname* to gain permitted access to a file. But neither a regular (hard) link nor a symbolic link changes the ownership, group, or permissions of a file. So even though you can make a link to a file that you can't read, you still won't be able to read its contents.

In the case of a hard link, you can remove either the original file or the link, and you are still left with a copy of the file. In the case of a symbolic link, if you remove the original file you are left only with the link; that is, a pointer to a file that does not exist anymore.

5 Editing and Writing With vi

5.1. What Is vi?

vi is the standard SunOS text editor ("vi" stands for "visual display editor"). Since vi is not window-based, it can be used on any kind of terminal.

You can enter and edit text with vi, but it is not a word-processor: you format vi text by inserting codes that are then interpreted by another program, a formatter. (Chapter 6 teaches you the basics of formatting a page using nroff and troff codes.)

In this chapter you'll learn many of the most useful vi commands. You'll find that it is a powerful text editor, but that it will take a little time in order to become proficient. More — and more technical — information can be found in "Using vi, the Visual Display Editor," in *Editing Text Files*.

Note that there is a read-only version of vi, called view. When you open a file with view, you can use vi commands, but you cannot write (or save) your changes.

5.2. Creating a File

In this section you'll learn how to start vi, enter text in a file, save (write) the file, and quit vi. A few simple commands will be introduced, and you'll make a practice file that you'll use for the rest of this tutorial. You'll also learn about the three *modes* of vi: command mode, last-line mode, and input mode.

Starting vi

vi is found in the directory /usr/ucb.

To start vi, type at a prompt the command vi, leave a space, type a filename, and then press Return.

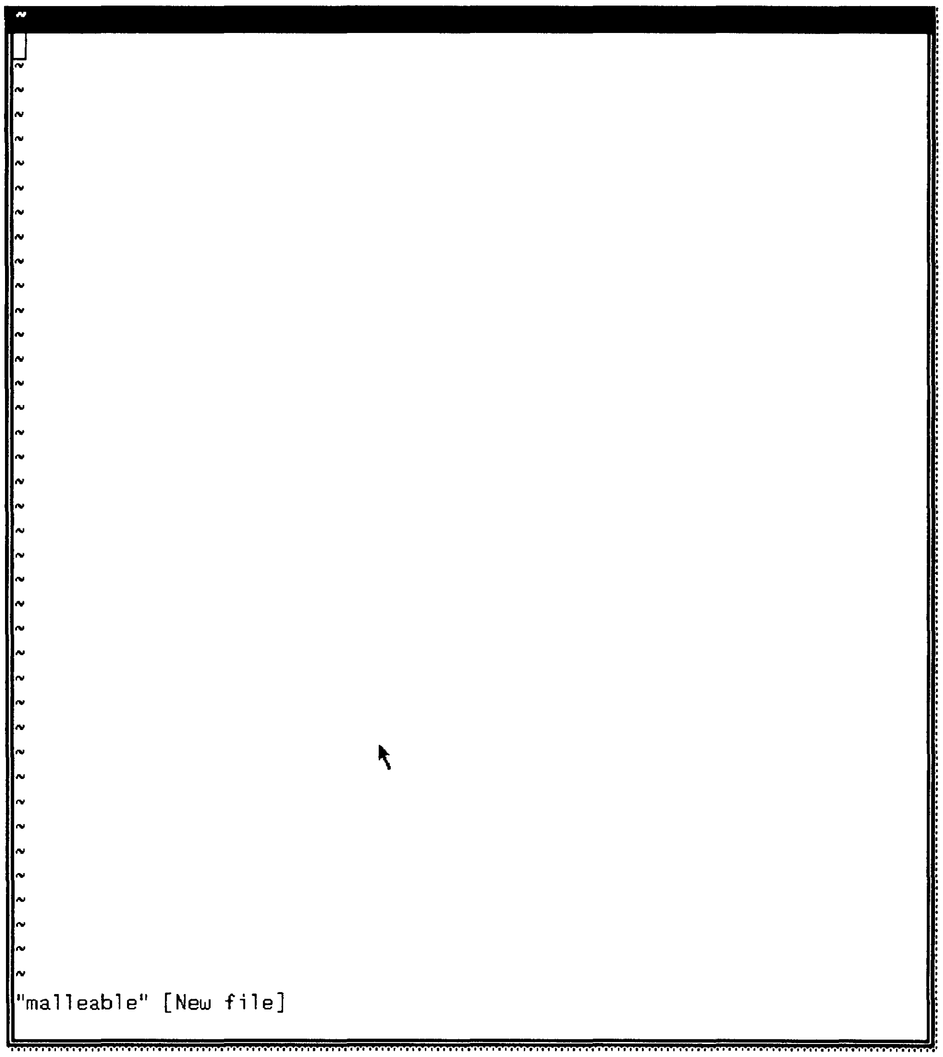
(If this file already exists, vi will open it; if this is a new file, vi will create it.)

For example, if you wanted to create a file called malleable, you would type:

```
venus% vi malleable
```

The vi screen will appear in a moment.

Figure 5-1 *New File*



```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"malleable" [New file]
```

The cursor appears in the upper lefthand corner of the screen. Blank lines are indicated by a vertical series of tildes (˜). The last line of the screen shows the name of the file and indicates that you're creating a new file.

Entering Text

One way to enter text in `malleable` is to type the letter `a`. This is a command meaning "append" — it won't show on the screen.

Now type a few short lines of text, ending every line with a [Return]. For the moment, you can correct your mistakes by backspacing and retyping a line *before* you press [Return]. Later you'll learn how to make corrections in more sophisticated ways.

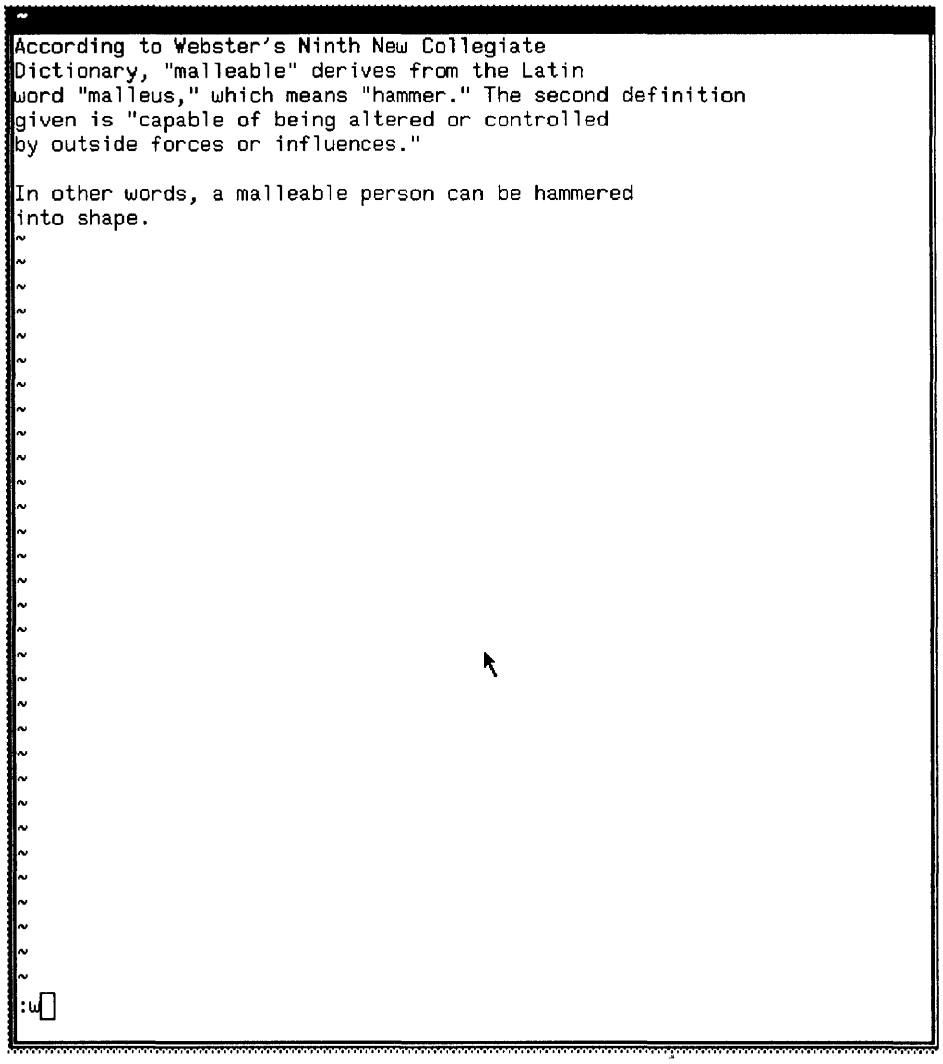
When you've finished entering text in `malleable`, type [Esc] ("escape"). The cursor will move back onto the last character entered.

Saving Your Work

Saving your work creates a record on disk of your file and any changes you've made to it.

To save the text you've entered in `malleable`, type Esc (to ensure that `vi` doesn't think you're still adding text). Now type a colon (:) followed by a `w` (for "write"). Press Return.

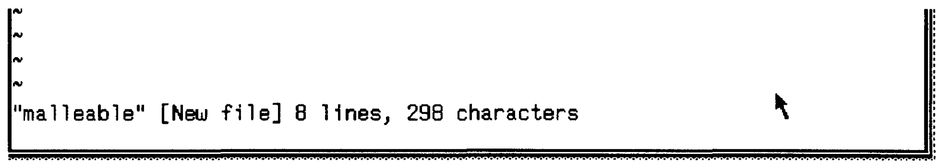
Figure 5-2 *Giving the Write Command*



```
According to Webster's Ninth New Collegiate
Dictionary, "malleable" derives from the Latin
word "malleus," which means "hammer." The second definition
given is "capable of being altered or controlled
by outside forces or influences."

In other words, a malleable person can be hammered
into shape.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:w
```

After you type Return, `vi` displays the filename, the number of lines in the file, and the number of characters in the file.

Figure 5-3 *File Saved*



```
~
~
~
~
"malleable" [New file] 8 lines, 298 characters
```

Quitting `vi`

Now that you've saved (or written) the contents of `malleable`, you can safely quit `vi`.

To quit `vi`, type a colon (:) followed by a `q`. Press [Return].

The system will display the command prompt to indicate that you have quit `vi`.

Printing a File

Unformatted files can be printed out with the command `lpr` *`filename`*. This command will send the file to your default printer. See Section 4.14 and Chapter 6 for information on printing files formatted with `nroff` and `troff`.

Some Theory: Command, Last-Line, and Input Modes

`vi` has three modes — commmand, last-line, and input. You have already used features of each: When you started `vi`, it came up in command mode. Typing `a` put you in input mode. Typing [Esc] took you out of input mode and put you back in command mode. Typing the colon in commands like `:w` or `:q` put you in last-line mode (also called "colon mode").

If `vi` seems to respond unpredictably, you may have inadvertently entered the wrong mode. You can always be sure that you're in command mode by pressing [Esc].

5.3. Moving Around in a File

In the previous section you learned how to create a file. Now that you have something to work with, you'll want to find out how to look at different parts of the file and how to move around.

Moving the Cursor

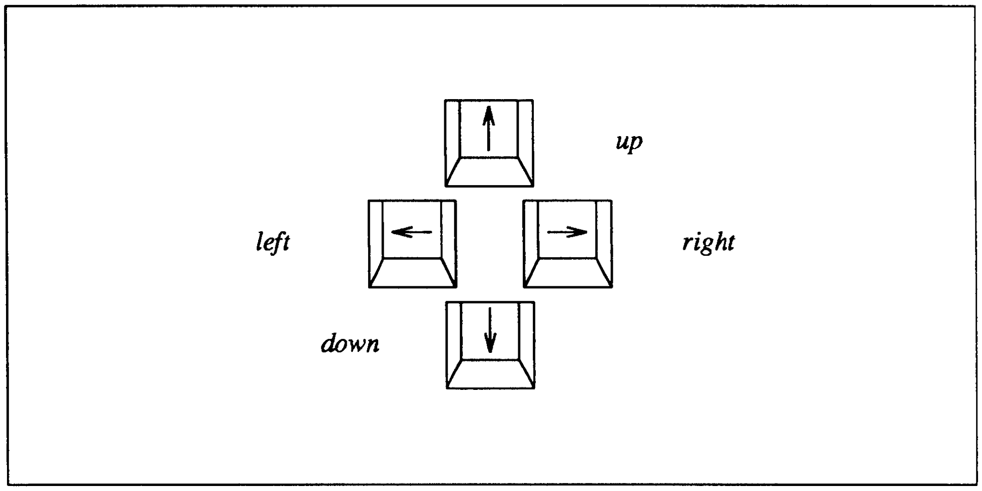
When you start `vi`, the cursor is in the upper lefthand corner of the `vi` screen. You can move the cursor with a number of keyboard commands. Some letters, the arrow keys, and the keys for [Return], [Back Space], and [Space Bar] can all be used to move the cursor when in command mode.

CAUTION **Most `vi` commands are case-sensitive: the "same" command typed in lowercase and uppercase could have radically different effects.**

Arrow Keys

If your machine is equipped with arrow keys, try these now. You should be able to move the cursor freely about the screen using combinations of the up, down, right, and left arrow keys. Of course, you can move the cursor only across already existing text or input spaces. (If you're using a remote terminal, however, the arrow keys may not work correctly, depending on your terminal emulator. Arrow key substitutes are discussed below.)

Figure 5-4 *Arrow Keys*



Arrow Key Substitutes: h, j, k, l

These keys work like arrow keys:

To move left: press `h`.

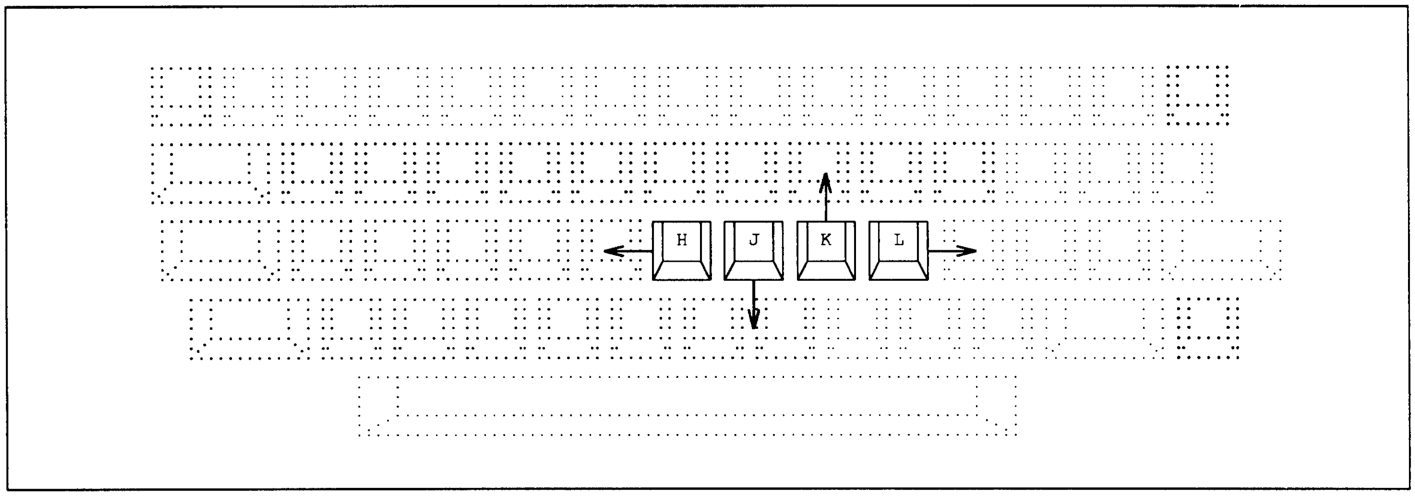
To move right: press `l`.

To move down: press `j`.

To move up: press `k`.

You can move a character or line at a time or, by holding down a key longer, move the cursor rapidly across the text.

Figure 5-5 *Cursor Moving Keys for* `vi`



One Word: w, b; W, B

Pressing w ("word") moves the cursor to the right a word at a time.

Pressing b ("back") moves the cursor left a word at a time.

Pressing W or B moves the cursor past the adjacent punctuation, to the next or previous blank space.

Down: Return

Pressing the [Return] key moves the cursor down.

Left: Back Space

Pressing the [Back Space] key moves the cursor left.

Right: Space Bar

Pressing the [Space Bar] moves the cursor right.

Top: H

Pressing H ("high") moves the cursor to the top of the screen.

Middle: M

Pressing M moves the cursor to the middle of the screen.

Bottom: L

Pressing L ("low") moves the cursor to the bottom of the screen.

Scrolling

You may have noticed that moving the cursor either to the bottom or top of the screen had the effect of scrolling text up or down. This may work okay for a very short file, but it's a tedious way to move through a long file.

By pressing the [Ctrl] key in combination with either `f, d, b,` or `u,` you can scroll through a file a screen or a half-screen at a time, backwards or forwards. (You might want to add text to `malleable` now to give yourself a longer file to experiment with.)

Scroll Forward: Ctrl-F

To scroll forward (i.e., move down) one screenful, press [Ctrl-F]. (By convention, uppercase letters are used to designate these combinations, but lowercase letters are used in practice.) The cursor will move to the upper lefthand corner of the new screen.

Scroll Down: Ctrl-D

To scroll down one half-screenful, press [Ctrl-D].

Scroll Backward: Ctrl-B

To scroll backward one screenful, press [Ctrl-B].

Scroll Up: Ctrl-U

To scroll up one half-screenful, press [Ctrl-U].

5.4. Inserting Text

`vi` has many commands for inserting text; this section introduces you to the most useful. By using a combination of cursor keys and the letters `a, A, i, I, o,` and `O`, followed by [Esc] ("Escape"), you'll learn how to insert text anywhere in a file. Note that each of these commands puts `vi` in input mode.

Append: a, A

Starting from command mode (remember: press [Esc] to be sure you're in command mode), you can insert text to the *right* of the cursor using `a`.

Experiment by moving the cursor anywhere on a line and typing `a`, followed by the text you want to add. Press [Esc] when you've finished.

Type `A` (uppercase!) to add text to the *end* of a line. To see how this works, position the cursor anywhere on a line and then type `A`. The cursor will move to the end of the line, where you can type your additions. Press [Esc] when you're done.

Insert: i, I

Insert text to the *left* of the cursor by typing `i` from command mode.

Type `I` to insert text at the *beginning* of a line. (The command will move the cursor from any position on a line.) Again, as with all the commands in this section, press [Esc] to return to command mode.

Open Line: o, O

These commands are used to open new lines, either above or below the current cursor position.

Typing `o` opens a line *below* the current cursor position. To experiment, type `o`, followed by a bit of text. Press [Esc] when you've finished.

Typing `O` opens a line *above* the current cursor position.

NOTE *Occasionally you may need to instruct* `vi` *to clear or redraw the screen to eliminate, e.g., extraneous system messages. Redraw the screen by entering command mode and pressing [Ctrl-L].*

5.5. Changing Text

Changing text involves substituting one bit of text for another. `vi` has several ways of doing this, depending on circumstances.

Change Word: cw

To replace a word, position the cursor at the beginning of the word to be replaced. Type `cw`, followed by the new word. To finish, type [Esc].

To change *part* of a word, place the cursor on the word, to the *right* of the portion to be saved. Type `cw`, the correction, and [Esc].

Change Line: cc

To replace a line, position the cursor anywhere on the line and type `cc`. The line will disappear, leaving a blank line for your new text (which can be of any length). Press [Esc] to finish.

Change Part of Line: C

Typing `C` allows you to replace *part* of a line: that part to the right of the cursor. Press [Esc] to finish.

Substitute Character(s): s

To substitute one or more characters for the character under the cursor, type `s`, followed by the new text. Press [Esc] to return to command mode.

Replace Character: r

This command is given to replace the character under the cursor with another character. To use it, position the cursor over a character and type `r`, followed by just one replacement character. After the substitution, `vi` returns automatically to command mode (there's no need to press [Esc]).

Transpose Characters: xp

Correcting transposed characters takes just two keystrokes in `vi`. Say you find that you've typed `teh` when you meant to enter `the` (this writer does it all the time!). Make the correction by putting the cursor over the first letter to be moved (in this case, `e`) and then giving the command: `xp`. The `e` and the `h` will trade places—and `vi` will automatically return to command mode.

Undo Previous Command: u

When making corrections, you'll sometimes wish you had *not* changed something. In `vi` you can "undo" your last command simply by pressing `u`. (Pressing `u` a *second* time undoes the "undo.")

Undo Changes to Line: U

Type `U` to undo *all* changes you've made to a line. This command will work only if you haven't moved the cursor off the line.

Break or Join Lines: r+Return, J

To break a line without affecting text, move the cursor to the breaking point (so to speak) and give the command `r` (for "replace") followed by Return.

To join two lines, place the cursor on the upper line and type the command `J`.

5.6. Deleting Text

These `vi` commands simply delete the character, word, or line you indicate. `vi` stays in command mode, so any insertions have to made with additional commands to enter input mode.

Delete Character: x

To delete one character, position the cursor over the character to be deleted and type `x`.

Using `x` to delete a letter will also delete the space the letter occupied— when a letter is removed from the middle of a word, the remaining letters will close up, leaving no gap. You can also delete blank spaces in a line with the `x` command.

Delete Word: dw

To delete a word, position the cursor at the beginning of the word and type the command `dw`. The word and the space it occupied will be removed.

To delete *part* of a word, position the cursor on the word, to the *right* of the part to be saved. Type `dw` to delete the rest of the word.

Delete Line: dd

To delete a line, position the cursor anywhere on the line and type the command `dd`. The line and the space it occupied will be removed.

Delete Part of Line: D

You can also delete part of a line — everything to the *right* of the cursor.

To do this, position the cursor to the left of the part of the line you want to delete and type `D`.

To learn how to delete text using last-line commands, see below.

5.7. Using Repeat Factors and Repeating Commands

Many `vi` commands can be preceded by a *repeat factor* — a number that precedes the command and tells it how many times to operate.

All the commands in the previous section take repeat factors. For instance, `3dd` would delete three lines, `2dw` would delete two words, and `4x` would delete four characters or spaces.

Typing a period (.) repeats the previous text-changing command. So if you've deleted a line with `dd`, you can move the cursor to another line and delete it by typing just a period.

5.8. Copying and Moving Text—Yank, Delete, and Put

Many word-processors allow you to "copy and paste" and "cut and paste" lines of text. Fortunately, `vi` follows suit in this regard. The `vi` command-mode equivalent of "copy and paste" is *yank and put*; the equivalent of "cut and paste" is *delete and put.* You can also copy and move text using last-line commands. Both methods are disussed below.

An easy way to copy or move small blocks of text is to use a combination of yank, delete, and put commands.

Copying Lines

To copy a line requires two commands, using `yy` or `Y` ("yank") and either `p` ("put below") or `P` ("put above").

`Y` is synonymous with `yy`.

To yank one line, position the cursor anywhere on the line and type the command `yy`. Now move the cursor to the line above where you want the yanked line to be put (copied). Type the command `p`. A copy of the yanked line will appear in a new line *below* the cursor.

You have the option of putting the yanked line in a new line *above* the cursor: type the command `P` to do this.

The `yy` command works well with a repeat factor: to yank 11 lines, for example, just type `11yy`. Eleven lines, counting down from the cursor, will be yanked, and `vi` will indicate this with a message at the bottom of the screen: `11 lines yanked`.

Moving Lines

Moving lines also requires two commands: `dd` ("delete") and either `p` or `P`.

To move one line, position the cursor anywhere on the line and type the command `dd`; to delete, e.g., five lines, type `5dd`.

Next, move the cursor to the line above where you want the deleted line reinserted. Type the command `p`.

Again, you can put the deleted line above the cursor by typing `P`.

CAUTION **Use only cursor-moving commands between yanking or deleting and putting. You can also use the search commands — `/`, `?`, `n` — discussed below.**

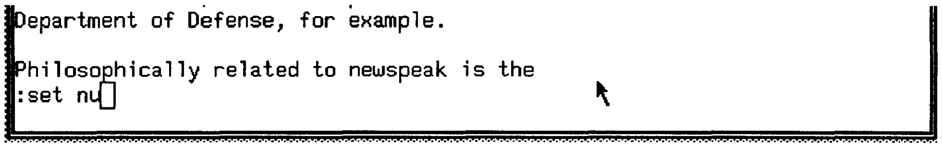
5.9. Copying, Moving, and Deleting Text— Last-Line Commands

Last-line commands are more accurate and convenient than yank, delete, and put when you're dealing with large blocks of text. Rather than counting lines on the screen and then searching for an insertion point, you give `vi` a range of lines to be moved or copied and then specify the line before the insertion point. (Of course, with the delete command there is no insertion point.)

Turning on Line Numbers

Turn line numbers on by typing the command `:set nu` and pressing (Return).

Figure 5-6 *Setting Line Numbers*



```
Department of Defense, for example.

Philosophically related to newspeak is the
:set nu
```

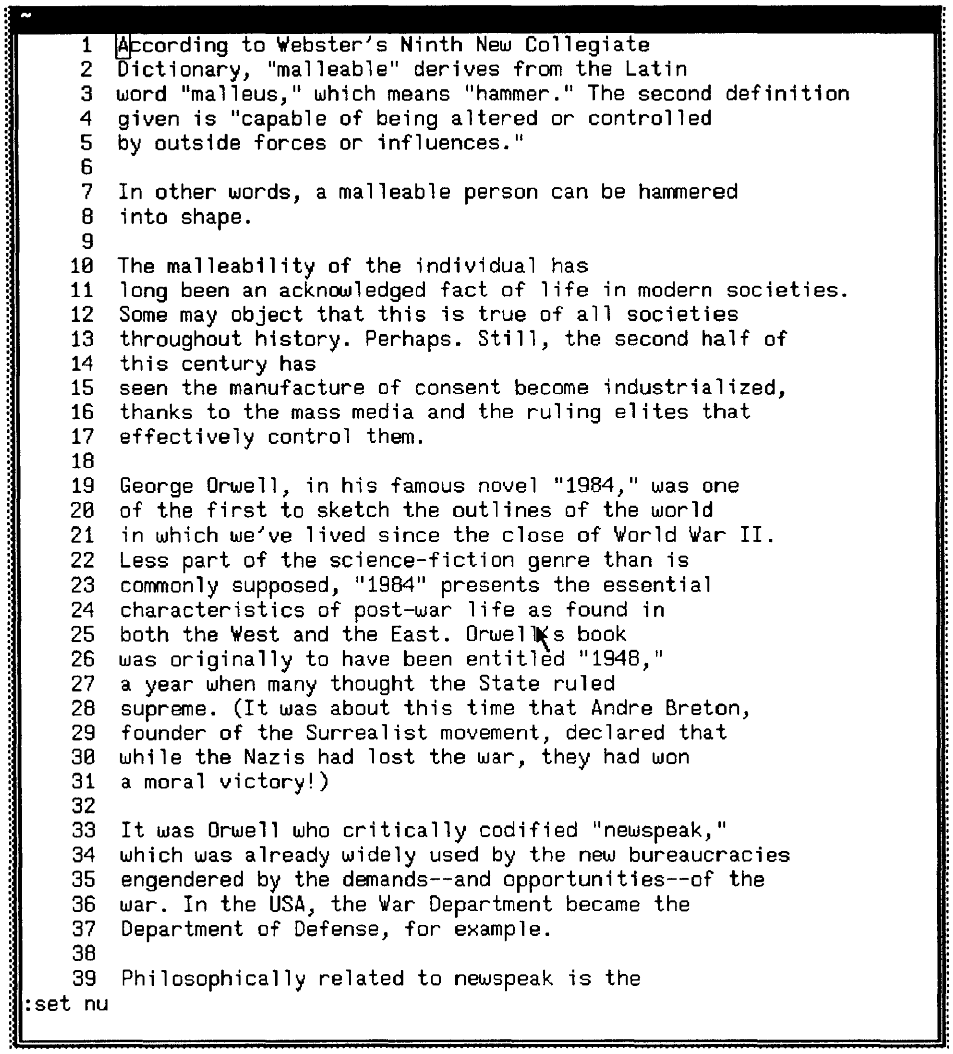
Line numbers will appear in the left margin. Note that these numbers do not print out on hard copy of a file. They are visible only on the screen.

Turn line numbers off by typing the command

```
:set nonu
```

and pressing (Return).

Figure 5-7 *Line Numbers Displayed*



```
      1  According to Webster's Ninth New Collegiate
      2  Dictionary, "malleable" derives from the Latin
      3  word "malleus," which means "hammer." The second definition
      4  given is "capable of being altered or controlled
      5  by outside forces or influences."
      6
      7  In other words, a malleable person can be hammered
      8  into shape.
      9
     10  The malleability of the individual has
     11  long been an acknowledged fact of life in modern societies.
     12  Some may object that this is true of all societies
     13  throughout history. Perhaps. Still, the second half of
     14  this century has
     15  seen the manufacture of consent become industrialized,
     16  thanks to the mass media and the ruling elites that
     17  effectively control them.
     18
     19  George Orwell, in his famous novel "1984," was one
     20  of the first to sketch the outlines of the world
     21  in which we've lived since the close of World War II.
     22  Less part of the science-fiction genre than is
     23  commonly supposed, "1984" presents the essential
     24  characteristics of post-war life as found in
     25  both the West and the East. Orwell's book
     26  was originally to have been entitled "1948,"
     27  a year when many thought the State ruled
     28  supreme. (It was about this time that Andre Breton,
     29  founder of the Surrealist movement, declared that
     30  while the Nazis had lost the war, they had won
     31  a moral victory!)
     32
     33  It was Orwell who critically codified "newspeak,"
     34  which was already widely used by the new bureaucracies
     35  engendered by the demands--and opportunities--of the
     36  war. In the USA, the War Department became the
     37  Department of Defense, for example.
     38
     39  Philosophically related to newspeak is the
:set nu
```

Copying Lines

The basic form of the last-line copy command is

```
:line#,line# co line#
```

The first two numbers (separated by a comma) specify the range of lines to be copied. The third number is the line *before* the insertion point.

For example, to copy lines 15 through 20 of `malleable` and place the copy after line 30, you would give the command

```
:15,20 co 30
```

and press [Return].

When specifying line ranges, you can use the abbreviations `.` (period) to denote "from cursor" and `$` (dollar sign) to denote "to end of file."

Thus, to copy the range "from cursor position through line 20" and insert this block after line 30, you would give the command `:.,20 co 30`. To copy the range "from line 15 through the end of the file" and insert this block after line 5, you would give the command `:15,$ co 5`.

Moving Lines

The basic form of the last-line move command is similar to the copy command discussed above:

```
:line#,line# m line#
```

Line ranges and insertion points are specified in the same ways, including use of the abbreviations `.` and `$`. The difference in function is simply that "move" deletes a block from one location and reinserts it elsewhere.

For example, to move lines 3 through 12 to the line following 21, you would give the command

```
:3,12 m 21
```

and press [Return].

You can delete a range of lines using the command form

```
:line#,line d
```

So, to delete lines 19 through 31, you would give the command

```
:19,31 d
```

NOTE *To "undo" last-line commands, give the command* `:u` *and press [Return]. Or just type* `u` *without first typing a colon.*

5.10. Searching and Replacing

`vi` provides several ways of finding your place in file (either by line number or by locating a string) or finding a specific bits of text and editing them. It also has a powerful replace function.

Finding a Character String

A *character string* is simply one or more characters in a row. It may include letters, numbers, punctuation, special characters, blank spaces, tabs, and carriage returns. A string may be a grammatical word or it may be part of a word.

To find a character string, type `/` followed by the string, and then press [Return]. `vi` positions the cursor at the next occurrence of this string.

Type `n` to go to the *next* occurrence of the string; type `N` to go to the *previous* occurrence.

You can type `?` instead of `/` to search backward in a file. In this case, the directions of `n` and `N` are reversed. In any event, `vi` searches in either direction wrap around the end of a file, looking for the string wherever it may occur.

Searches normally are case-sensitive: a search for "china" will not find "China."

To look for a string in `malleable`, first type / and then type a common word, `disinformation`, for example. Press [Return].

```
/disinformation
```

If `vi` finds the string, the cursor will stop at its first occurrence. If the string is not found, `vi` will display `Pattern not found` on the last line of the screen.

Refining the Search

For the advanced user:

You can make searches more precise by tagging the string with indicators for, among other things, "beginning of line," "end of line," "beginning of word," and "end of word." You can also generalize a search by using wild-card characters. (More information on these special search characters can be found in the section on `grep` in Section 7.11.)

To match *the beginning of a line,* start the search string with a caret (^). To find the next line beginning with "Disinformation," give the command

```
/^Disinformation
```

To match *the end of a line,* end the search string with a dollar sign ($). To find the next line ending with "disinformation.", give the command

```
/disinformation\.$
```

(Note that the period is quoted with a backslash.)

To match *the beginning of a word,* type \< at the beginning of the string; to match *the end of a word,* type \> at the end of the string. Thus, to match *a word,* combine the end-of-word and beginning-of-word tags in the search pattern. To find the next occurrence of the word—as opposed to the string—"disinformation," give the command

```
/\<disinformation\>
```

To match *any (unknown or variable) character,* type a period (.) in the string. To find the next occurrence of "disinformation" or "misinformation," give the command

```
/.isinformation
```

Because this is a string, and not a word, this search pattern will also find "misinformationalist" and "disinformationism."

Brackets may be used to search for alternative characters in a string. `/[md]`*string* will find strings beginning with *either* "m" *or* "d." On the other hand, `[d-m]`*string* will find strings beginning with any letter *from* "d" *through* "m."

To match *zero or more occurrences of the last character,* type an asterisk (*) in the string. You can effectively combine brackets and the asterisk to look for well-defined alternatives. To find all strings beginning with "a" through "z" and ending with "isinformation" *and* to find all occurrences of the string "isinformation," give the command

```
/[a-z]*isinformation
```

Replacing a Character String

The procedure for replacing a text string is based on the search procedures discussed above. All the special matching characters for searches may be used in search-and-replace.

The basic command form is

`:g/`*`search-string`*`/s//`*`replace-string`*`/g`

followed by [Return].

Thus, to replace every occurrence of the string "disinformation" with "newspeak," you would give the command

```
:g/disinformation/s//newspeak/g
```

and press [Return].

You can modify this command to halt the search and make `vi` query whether you want to make the replacement in each instance. The commmand `:g/disinformation/s//newspeak/gc` (adding the `c` for "consult") will make `vi` stop at every occurrence of "disinformation" and ask whether you want to make the substitution. Respond with `y` for yes or `n` for no.

NOTE *You can cancel a "consulted" search-and-replace by pressing* [Ctrl-C].

Finding a Specific Line

`vi` provides ways of accessing parts of a file by either opening the file to a specific line or by finding a line in an already opened file.

Going to a Line With G

You can go to the last line of an open file by typing `G`. Return to the first line of the file by typing `1G`.

You can go to any other line by typing its number followed by `G`.

For example, say you've quit the `malleable` file while editing line 51. You can access that line by opening the file and typing `51G`.

Opening a File at a Specific Line

You can also start vi and open a file at a specific line. At a command prompt, type

```
vi +line# filename
```

To start and open malleable at line 51, give the command

```
venus% vi +51 malleable
```

(By not supplying a line number after the plus sign, you can make vi open malleable at the *last* line of the file.)

Opening a File at a Pattern

You can open a file to the first line containing a specific string by typing at a command prompt

```
vi +/pattern filename
```

For example, to open malleable at the first line containing the string "2+2=5," you would give the command

```
venus% vi +/2+2=5 malleable
```

and press Return.

Enclose multiple-word strings in double quotation marks, e.g.:

```
venus% vi +/"Andre Breton" malleable
```

5.11. Inserting a File Into a File

vi makes it convenient to "read" (insert) a file into the file you're editing. The general form of the command is

```
:line# r filename
```

If you do not specify a line number, vi puts the file at the current cursor position.

If you were working on the file malleable and you wanted to read in another file called orwell, and you wanted it placed at line 84, you would give the command

```
:84 r orwell
```

Or you could position the cursor on line 84 and type

```
:r orwell
```

5.12. Ending a Session

Using the Buffer

When writing or editing a file in `vi`, your changes are not made directly to the file. Instead, they are applied to a copy of the file that `vi` creates in a temporary memory space called the *buffer*. The permanent disk copy of the file is modified only when you *write* (save) the contents of the buffer.

This arrangement has its good and bad points. On the one hand, it means that you can quit a file and discard all the changes you've made during an editing session, leaving the disk copy intact. On the other hand, you could lose the (unsaved) contents of the work buffer if the system crashes. (People on remote terminals connected by phone lines are especially vulnerable to unplanned interruptions.)

Probably the best policy is to save your work frequently, especially when making substantive changes.

CAUTION **Although it's possible to run multiple, simultaneous `vi` sessions on one file, it is *not* a good idea. Great confusion could result when you try to determine which changes have been written to the file.**

Saving Changes and Quitting `vi`

`vi` is rich in more or less synonymous commands that control saving the buffer contents to a file and quitting `vi`. These commands give you the option of saving, saving-and-quitting, or quitting-without-saving.

Saving

Save the contents of the buffer (or, write the buffer to the file) by giving the command

```
:w
```

followed by [Return].

Saving and Quitting

Save and quit by giving the command

```
:wq
```

Alternatively, type `ZZ`. Note that this command is neither preceded by a colon nor followed by [Return].

Quitting Without Saving

When you've made no changes to a file and simply want to quit, give the command

```
:q
```

If you have made changes, `vi` will not let you quit. Instead, it will display the message, `No write since last change (:quit! overrides).`

If you do not want to save your changes, give the command

```
:q!
```

NOTE *You can "suspend" or interrupt* `vi` *by typing* Ctrl-Z. *A command prompt will appear. To return to* `vi`, *type* `fg` *at the command prompt. Press* Return.

Recovering From a Crash

If the system crashes, the contents of your buffer are at risk. Often, though, you can recover most of your work by restarting `vi` with the command form

```
vi -r filename
```

The system will usually send you mail about this. You would recover `malleable` by giving the command

```
venus% vi -r malleable
```

5.13. Customizing vi

`vi` has many variables that affect its behavior and appearance. You can view a list of these variables (with their current settings) by giving the command

```
:set all
```

followed by Return. Refer to "`vi` and the `.exrc` File" in *SunOS User's Guide: Customizing Your Environment* for information on how to set the options you want as defaults every time that you start `vi`. "Using `vi`, the Visual Display Editor," in *Editing Text Files,* discusses these options in detail. That chapter also contains information on how to set up the many different display terminals on which `vi` runs.

Type at a prompt **`man vi`** for additional information.

5.14. Problems With vi

If you run into difficulties running `vi`, *SunOS User's Guide: Basic Troubleshooting* offers some tips on handling minor problems.

5.15. A Note on ex

`vi` is a superset of `ex`, a line-oriented text editor. `ex` commands can be combined with `vi` commands. In fact, you've already used some: commands preceded by a colon are examples of `ex` commands (`:w`, `:q!`). For more information, see "Command Reference for the `ex` Line Editor," in *Editing Text Files.*

Table 5-1 *Basic* `vi` *Commands*

Basic `vi` Commands	
Starting `vi`	
`vi` *filename*	open or create file
`vi +18` *filename*	open file to line 18
`vi +/"mustard greens"` *filename*	open file to first occurrence of "mustard greens"
`vi -r` *filename*	recover crashed file
`view` *filename*	open file read-only
Cursor Commands	
`h`	move left
`j`	move down
`k`	move up
`l`	move right
`w`	move right one word
`W`	move right one word (past punctuation)
`b`	move left one word
`B`	move left one word (past punctuation)
`Return`	move down one line
`Back Space`	move left one character
`Space Bar`	move right one character
`H`	move to top of screen
`M`	move to middle of screen
`L`	move to bottom of screen
`Ctrl-F`	scroll forward one screen
`Ctrl-D`	scroll forward one-half screen
`Ctrl-B`	scroll backward one screen
`Ctrl-U`	scroll backward one-half screen
Inserting Characters and Lines	
`a`	insert characters to right of cursor
`A`	insert characters to right of cursor, at end of line
`i`	insert characters to left of cursor
`I`	insert characters to left of cursor, at beginning of line
`o`	insert line below cursor
`O`	insert line above cursor
Changing Text	
`cw`	change word (or part of word right of cursor)
`cc`	change line
`C`	change part of line to right of cursor
`s`	substitute string for character under cursor
`r`	replace character under cursor with one other character
`r-Return`	break line
`J`	join current line and line below
`xp`	transpose character at cursor & character to right
`~`	change case of letter (upper or lower)
`u`	undo previous command
`U`	undo all changes to line
`:u`	undo previous last-line command

Table 5-1 *Basic* vi *Commands— Continued*

Basic vi Commands	
Deleting Text	
`x`	delete character
`dw`	delete word (or part of word to right of cursor)
`dd`	delete line
`D`	delete part of line to right of cursor
`:5,10 d`	delete lines 5-10
Copying and Moving Text	
`yy`	yank or copy line
`Y`	yank or copy line
`dd`	delete line
`p`	put yanked or deleted line below current line
`P`	put yanked or deleted line above current line
`:1,2 co 3`	copy lines 1-2 and put after line 3
`:4,5 m 6`	move lines 4-5 and put after line 6
Setting Line Numbers	
`:set nu`	show line numbers
`:set nonu`	hide line numbers
Finding a Line	
`G`	go to last line of file
`21G`	go to line 21
Searching and Replacing	
/*string*/	search for *string*
?*string*?	search backward for *string*
`n`	find next (or previous) occurrence of *string*
:g/*search-string*/s//*replace-string*/gc	search and replace, consult at each occurrence
Clearing the Screen	
`Ctrl-L`	clear scrambled screen
Inserting a File Into a File	
:r *filename*	insert (read) file after cursor
:34 r *filename*	insert file after line 34
Saving and Quitting	
`:w`	save changes (write buffer)
:w *filename*	write buffer to file
`:wq`	save changes and quit vi
`ZZ`	save changes and quit vi
`:q!`	quit without saving changes

6

Formatting and Printing Documents

Text editors like `vi` produce ``raw,'' unformatted files. A *text formatter* lets you specify concisely the way you want your printout to look. You can add special attributes, like different fonts and intensities, that would be impossible without the formatter.

In this chapter, you will learn the basics of the SunOS formatters called `nroff` and `troff`. `nroff` is used to produce typewriter-like output for conventional dot-matrix and letter-quality printers, though it may be used for laserprinters. While `troff` was developed to produce output for phototypesetters, it's mainly used now to format text for laserprinters. All the commands discussed in this chapter work with both programs. The versions of `nroff` and `troff` discussed here reside in your `/usr/bin` directory.

6.1. Sample Memo

To help you learn `nroff` and `troff`, we've included two forms of a sample memo. First you'll see the printed version of the memo, as it would come out of a laserprinter. Then you'll see the file as you would input it on your screen, with the formatting commands visible. Compare the two versions line for line to see how the `nroff` and `troff` commands affect the final copy.

From: Board of Directors
To: Vociferous Employees

Cut Paper Waste on Memos

It has come to our attention that certain employees of this company have been printing inordinate numbers of inter-office memos. As you know, our distinctive Flying Headstone stationery is quite expensive.

We have decided to take three immediate steps to counter this **threat** to corporate profitability:

1) We will reduce paper stocked for the printers by 25% within month-end.

2) We will begin terminating employees randomly.

3) We will open fire on anyone caught within 100 yards of a printer.

If these steps do not stop the proliferation of memos, we will consider sending out an even stronger note on the subject.

Here is the sample memo before formatting. All of the lines that begin with a period are `nroff` and `troff` formatting commands.

```
.LP
From: Board of Directors
.br
To: Vociferous Employees
.sp
.ce
.I "Cut Paper Waste on Memos"
.LP
It has come to our attention that certain employees
of this company have been printing
.UL inordinate
numbers of inter-office memos.  As you know, our
distinctive Flying Headstone stationery is quite
expensive.
.LP
We have decided to take three immediate steps to
counter this
.B threat
to corporate profitability:
.IP 1)
We will reduce paper stocked for the printers by 25% within
month-end.
.IP 2)
We will begin terminating employees randomly.
.IP 3)
We will open fire on anyone caught within 100 yards of a
printer.
.LP
If these steps do not stop the proliferation of memos,
we will consider sending out an even stronger note on
the subject.
```

6.2. Basic Formatting Commands

The commands in lowercase letters are called *primitives*: they are the simplest commands, directing the formatter to break a line, add a space, or center the text. The uppercase commands are called *macros*: they are preprogrammed combinations of primitives that tell tell `nroff` and `troff` to do more complicated things, like create an itemized paragraph.

The macro package used in this chapter is called `-ms`.

`.LP`	to left-justify a paragraph
`.IP`	to create an itemized paragraph (like this one)
`.I`	to *italicize* text in `troff` (or to underline in `nroff`)
`.UL`	to <u>underline</u> text in both `troff` and `nroff`

`.B`	to make text **boldface** in `troff` (or to strike over in `nroff`)
`.ce`	to center text on the page
`.sp`	to create a blank line-space
`.br`	to force the end of a line, a line-break

Left-Justified Paragraph

Use the `.LP` command to begin a paragraph without indentation. `.LP` must appear all by itself on the line before the paragraph.

Itemized Paragraph

The `.IP` command begins an itemized paragraph. An itemized paragraph starts with an item, say the figure `1`, followed by an indented paragraph of text.

Put `.IP` on its own line, followed by the item you want to mark the paragraph. The itemized paragraphs in the sample memo are enumerated paragraphs, but you can also begin itemized paragraphs with other items.

Italicizing Text

To italicize a word, type the `.I` command at the beginning of a line, followed by the word on the same line. (This command underlines in `nroff`.) If the text to be italicized is more than one word long, enclose the text in double-quotation marks (" ").

Underlining Text

To underline text, type the `.UL` command at the beginning of a line, followed by the text you want underlined. If the text to be underlined is more than one word long, enclose the text in double-quotation marks (" ").

Boldface Text

To boldface text, type the `.B` command at the beginning of a line, followed by the text on the same line. (In `nroff`, the the printer will attempt to strike over the text.) If the text to be made boldface is more than one word long, enclose the text in double-quotation marks (" ").

Centering Text

When you have a title or headline to center, type the `.ce` command on a line by itself, then type the text you wish to center on the next line.

Line-Spacing

`.sp` creates an empty line, or line-space, in the file. Type `.sp` on a line by itself.

Line Breaks

When you want a line to end at a specific point, type `.br` on a line by itself.

6.3. Formatting and Printing the File

Once text has been input and coded, it's time to run the `nroff` and `troff` formatters on the file and print it.

Using `nroff`

To run the `nroff` formatter, give the following command:

```
venus% nroff -ms sample.memo > sample.ms
venus%
```

`sample.ms` is the `nroff`-formatted file. (People often name formatted files after the macro package.)

You can preview the formatted file on screen by using the `more` program:

```
venus% more sample.ms
```

To print the formatted memo, type `lpr` followed by the filename of the formatted file:

```
venus% lpr sample.ms
venus%
```

Alternatively, you can format and print without creating an intermediary file. You do this by running `nroff` on the file and then piping the output to the printer command:

```
venus% nroff -ms sample.memo | lpr
venus%
```

For more on pipes, see Chapter 7.

To preview, pipe the command to `more` instead of to `lpr`. Then, when the document is ready to print, repeat the command and pipe it to `lpr`.

Using `troff`

Format and print your file by running `troff` on the file and then piping the output to the printer command:

```
venus% troff -ms -t sample.memo | lpr -t
venus%
```

No preview function is supplied for `troff`-formatted files.

6.4. Where To Find Out More About Formatting

See *Using* `nroff` *and* `troff`, *Formatting Documents*, and the `nroff` and `troff` man pages for more information.

7

Basic Command Syntax

7.1. SunOS Commands: An Overview

The SunOS operating system may seem complex and arcane; it is, however, very flexible and powerful. It allows you to create your own commands, devise shortcuts for repetitive tasks, and perform complex tasks quickly and easily.

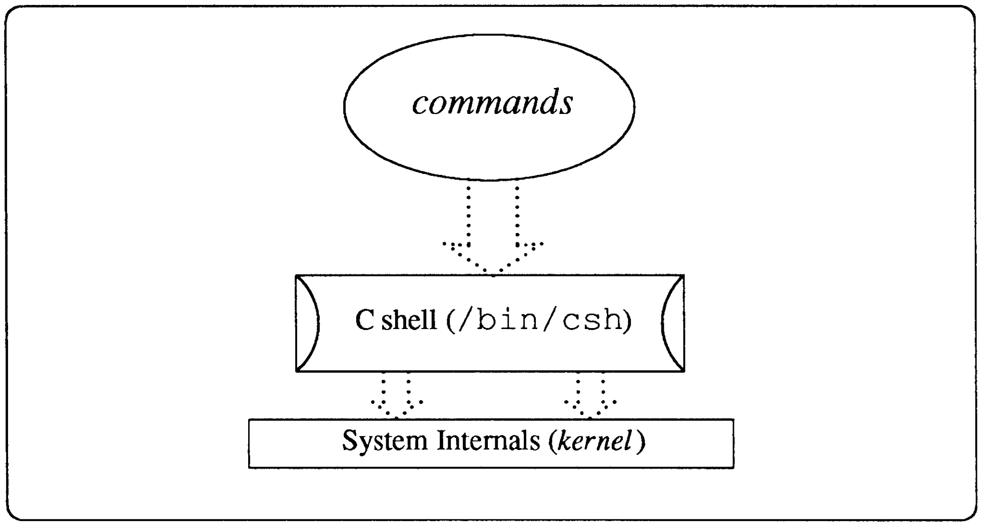
Before you read this chapter, you should have a basic knowledge of the SunOS file system and how to manipulate it; this information is contained in Chapters 3 and 4.

This chapter introduces the command syntax of the SunOS C shell. The C shell is a *command interpreter* —it deciphers and executes the commands you put together.

Another shell—the Bourne shell—is also available. See *SunOS User's Guide: Doing More.*

You can think of the C shell as a layer of software between you and the internal workings (or kernel) of the system.

Figure 7-1 *Where the C Shell Sits*



This tutorial combines C shell commands with SunOS utilities such as `cat`, `sort`, `head`, `tail`, `ps`, `at`, `batch`, and `grep`. These essential concepts of shell use are discussed and illustrated:

- Parts of the command line;

- Filename matching;
- Redirecting standard input and output;
- Redirecting standard error;
- Filters;
- Command line editing with the history mechanism;
- Processes;
- Aliases;
- Running commands in the background; and
- Searching for patterns.

You can write simple programs for the C shell, to do many types of tasks. *SunOS User's Guide: Doing More* explains how these programs, called "shell scripts," work. *SunOS User's Guide: Doing More* also continues the discussion of the history mechanism and demonstrates the use of variables in the C shell. Additonal information on the C shell can be obtained by typing **man csh** at a prompt.

7.2. Starting a C Shell

A C shell is started whenever you log in or give the commands `shelltool` or `cmdtool` ('Shell Tool' or 'Command Tool' in the SunView menu). A subshell (a shell within a shell) is started with the command `csh`. Type `exit` to escape a subshell.

7.3. Arguments and Options

A command line is composed of a command and *arguments*.

Consider this example:

```
venus% ls -l *.ps
```

This command line consists of a command and two arguments. There are several kinds of arguments: they may be be options, objects, patterns, redirection symbols, or even other commands.

Here's a breakdown of this command line:

`ls` A *command*, to list contents of directories.

`-l` An *option*, modifying the behavior of `ls`.

`*.ps`
The *object* of the command—in this case, the files `ls` is to display.

Every argument is associated with a command. In our example command line, everything to the right of `ls` can be considered an argument of `ls`.

Options

Certain arguments, like `-l`, are called *command options*. They modify the behavior of a command. Each command has its own unique options (or it may have none at all). If you look at the page on `ls` in the *SunOS Reference Manual*, you'll see that `ls` has 20 or so options. (Some of these are described in detail following Section 4.7.)

Options typically start with a dash (-).

7.4. Matching Filenames

The abbreviations (or "metacharacters") `?` and `*` are explained in Chapter 4. Briefly, `*` matches one or more characters in a file's name, and `?` matches any single character.

```
venus% ls ma*
maine          maple          marsh
manitoba       maryland
venus% ls ma???
maine          maple          marsh
venus%
```

Other Syntaxes for Matching Filenames

In addition to the wild cards `*` and `?`, the SunOS operating system provides more sophisticated ways of specifying a set of files on the command line.

Single-Character Matching with [and]

You can use *brackets* instead of a `?`, to match a single character. Within the brackets you can specify a list of characters to match against. For instance,

```
[ab]*
```

matches all filenames that begin with a lowercase `a` or `b`. You can also specify a *range* of characters to match against. Thus,

```
[A-Z]*
```

matches all filenames that begin with an uppercase alphabetical character.

You can use *braces* instead of `*` to match specific character strings of any length. Within the braces, strings are separated by commas. For instance,

```
{uranus,sygnus,x}*
```

matches any filenames beginning with `uranus`, `sygnus` or `x`.

Within braces, `*` and `?` are legal. You can nest braces within strings for interesting results. For instance, `{{ura,syg}nus,x}*` is another way to match filenames beginning with `uranus`, `sygnus` or `x`.

7.5. Command Sequences

Separating Commands With a Semicolon

You can combine several commands on one line by separating them with a semicolon (`;`).

```
venus% cd /usr/spool ; pwd
/usr/spool     (pwd prints out current directory)
venus%
```

Continuing Long Command Lines

If a command line is longer than the width of the screen you're typing it in, you can break it up with a backslash (\).

```
venus% /bin/rm xoutline; egrep -h "^.FN|^.TN|^.H" \
preface `cat Listfile` > xoutline
venus%
```

7.6. Redirecting Input and Output

Unless you indicate otherwise, commands normally display their results on the terminal screen. In this case, the terminal is known as the command's *standard output.*

Also, commands normally operate on data as you type it in from the keyboard; normally, then, the terminal is the command's *standard input.*

Because SunOS commands treat files and devices in a uniform way, you can direct the output of a command to any file or device that you choose. You can also use the output of one command as direct input to another, using a special connection symbol called a *pipe* (explained later). Or you can obtain the input to a command from a file.

Redirecting the Standard Output

A right *angle-bracket* (>) (pronounced "into") on the command line indicates that the next word is the name of a file or device in which to place, or *redirect* the output of a command. For instance, the command line:

```
venus% ls > list
venus%
```

places the output of the `ls` command (a list of files) in a file named `list`.

CAUTION **If a file by that name already exists, any previous contents may be deleted *before* the command is performed.**

So the command

```
cat file1 > file2
```

removes all existing contents from `file2` before the `cat` command is executed.

To avoid writing over existing files, add a line with the command

```
set noclobber
```

to your `.cshrc` file, if one isn't there already. (Refer to *SunOS User's Guide: Customizing Your Environment* for more information about this file.) Then, to make changes effective now, type in the command:

If using windows, type this `source` command in each `shelltool` or `cmdtool` window, so that the change will take effect in each.

```
venus% source ~/.cshrc
venus%
```

When you are certain that you want to overwrite the previous contents of a file, use >! to override this file protection.

```
venus% cat file1 >! file2    (file2 is overwritten)
venus%
```

Appending to an Existing File

You can *append* to the end of a file using a *double right angle-bracket* (>>) (pronounced "onto"). For example, in the example below, we add the output of the `pwd` command (showing the current directory) to the file `list`, created above.

With `noclobber` set, a file must already exist before the standard output can be appended to it. Using >>! overrides this.

```
venus% ls > list    (create list)
venus% cat list    (look at it)
SCCS
csh.1
files
list
outline
philosophy
picture1
preface
venus% pwd >> list    (append the output of pwd)
venus% cat list    (look at list again)
SCCS
csh.1
files
list
outline
philosophy
picture1
preface
/home/venus/medici
venus%
```

Redirecting Input

Just as you can redirect the output of a command, you can also specify a file (or device) from which that command obtains its *input.*

You can use a *left* angle-bracket (<) (pronounced "from") to redirect the standard input of a command. For instance, in the example above we used the command

```
cat list
```

to show the contents of the file `list`; another way would have been to say

```
cat < list
```

Most commands, like `cat`, allow the input file to be specified as an argument. However, other commands, such as `crypt`, only read from the standard input and thus require the use of <.

Pipes and Pipelines

The output of one command can be fed in directly as input to another. A set of commands strung together in this way is called a *pipeline*. The symbol for this input/output (I/O) connection is a vertical bar (|) (pronounced "through"), called a *pipe*. Pipes and pipelines have a wide variety of uses.

As an example, suppose you wanted to know how many files there are in a directory. You could use `ls` to display the contents of the directory, and then hand-count the number of files it shows. An easier way is to pipe the output of `ls` through the command `wc`, a word-counting program.

`wc` displays the number of lines, words, and characters, respectively, of the input given it. Here's how you'd count the number of files in the directory `/etc`:

```
venus% ls /etc | wc
      86      86     684
venus%
```

As you can see, the example displays only the output of `wc`; the output of `ls` feeds into `wc`. `wc` itself tells us that the output of `ls` contained 86 lines, 86 words, and 684 characters. Since each file makes up one word of the output of `ls`, we know that `/etc` contains 86 files. (`ls` displays only one file to a line when the output doesn't go directly to the screen.)

You can connect several commands to make longer pipelines. For instance, the command line:

```
venus% ls /etc | grep '^re.*' | wc
       4       4      32
```

displays the number of files in `/etc` starting with "re" (four). (The `grep` command is explained in Section 7.11.)

Filters

Commands like `wc` are called *filters*. They accept text as input, transform or analyze it, and produce text as output. Although often used as commands in their own right, filters are especially useful in pipelines.

`ls` is not a filter, because it doesn't accept data from the standard input. That is, you can't give `ls` any data to work on, because it gets its information from the directory structure.

The `date` command isn't a filter, either, even though you can give it data (i.e, you can set the date). Why not? Because you can only give it information by way of command-line arguments.

As you might expect, the command

```
ls | date
```

produces *only* the date, since `date` ignores its standard input. And

```
date | ls
```

produces only the output of `ls`.

Here are some other things you can do with filters:

display data one screenful at a time (`more`)
As explained in Section 4.9, the `more` command displays a file in screen-size chunks. This also works for output sent to `more` from another command.

search for specific patterns (`grep`)
`grep` allows you to look for just the output lines which are of interest to you. `grep` is covered in detail in Section 7.11.

display the first *n* lines of a file (`head`)
The `head` command shows the first lines of a file; you can specify how many lines you want displayed. (The default is ten.) `head` is useful for quickly comparing similarly named files.

Here we compare the first four lines of three files:

```
venus% cat speech* | head -4
==>speech1<==
Fourscore and seven years ago--or, say,
seven dozen and three years ago--or, for those
in the audience who like Base Two, forty-three
and one-half pairs of years ago--our forefathers
==>speech2<==
I'm reminded of the famous story of the two
Hungarian shoe salesmen and the Emperor penguin.
It seems that two Hungarian shoe salesmen were
in Antarctica--I forget why they were in Antarctica,
==>speech3<==
Ladies and gentlemen, I have a dream.  It is a
dream deeply rooted in the American dream.  In
it I am flying over my high school, and it's the
last day of classes, and I've forgotten to study,
venus%
```

display the last *n* lines of data (`tail`)
Similar to `head`, `tail` displays the last *n* lines. With no *-n* argument, it displays the last ten.

```
tail +n
```

skips to line *n* and displays that line through the end of the file.

display data, finding a search pattern (`more +`)
The command

```
more +/pattern
```

begins displaying data two lines before the first match for *pattern*, which can be either a string or a `grep` search pattern (described in 7.11). `more` is also described in Section 4.9.

display nonprinting characters (`cat -v`)

```
cat -v
```

translates nonprinting characters into strings of regular characters of the form *^c* (for control characters), or `M-`*c* (for 8-bit characters).

sort lines in predetermined order (`sort`)
The `sort` command sorts and collates lines according to an order you specify. Refer to `sort` in the *SunOS Reference Manual* for more information. For example,

```
sort -n
```

sorts in numerical order.

format text (`fmt`)
`fmt` does rudimentary formatting of text. For example, you can make all your lines in a file, or in a text window, 72 characters long.

reverse order of characters in a line (`rev`)
`rev` reverses the order of characters within each line.

reformat input (`pr`)
`pr`, like `fmt`, reformats input.

`lpr` is the command for printing files.

`pr` is useful for preparing files for printing. For example,

```
cat filename | pr -t -n | lpr
```

breaks up the output into *n* columns. The `-t` option suppresses a heading that would otherwise appear. There are a number of options to `pr`; see the *SunOS Reference Manual* for more information.

check spelling (`spell`)
`spell` produces a list of (possibly) misspelled words.

perform simple editing on files or output (`sed`)
`sed` uses *regular expressions* to represent patterns of text. (Regular expressions and `sed` are explained in *Editing Text Files*.)

Suppose you're using the `grep` utility to find instances of the word "butter" in various files. (`grep` is explained in Section 7.11.)

```
venus% grep "butter" chap*
chap1:the world's butter supply in danger, we
chap8:butter and motor oil to make a delicious
chap30:life as we know it without butter would be
chap112:primitive butter-worshiping peoples in Central
venus%
```

You can use `sed` to modify the output to make it easier to read:

```
venus% grep "butter" chap* | sed "s/:/: /"
chap1:  the world's butter supply in danger, we
chap8:  butter and motor oil to make a delicious
chap30: life as we know it without butter would be
chap112:    primitive butter-worshiping peoples in Central
venus%
```

Redirecting the Standard Error

When a command performs without problems, it produces results on its standard output. But when a command encounters a problem, it uses a different channel to send error messages to the terminal. This second channel, called the *standard error,* can also be redirected.

You can redirect the standard error to the same destination as the standard output by appending an ampersand (`&`) to the output-redirection symbol.

`>&` sends both standard and diagnostic output to a destination file. `>>&` appends the output to the file. `|&` includes both types of output as input to the next command in the pipeline.

If you want a command to perform silently, that is, to display no output of either kind, you can redirect its output to `/dev/null`, the system "wastebasket."

```
command >& /dev/null
```

To separate the standard error from the standard output, use a command line of the form:

```
(command > outfile) >& errorfile
```

When you want to force output to appear on the terminal, you can redirect it to `/dev/tty` (a synonym for the name of the terminal).

```
command >& /dev/tty
```

So the command

```
venus% (program > /dev/null ) >& /dev/tty
```

throws away the standard output and displays only the error messages produced by `program` (if any). This construction can save you time when testing long-running commands.

Table 7-1 *Summary of Redirection Symbols*

>	send output to named file
>!	same, overwrite file if it exists
>&	send standard error to file
>>	append output to file
>>!	same, overwrite file if it exists
>>&	append standard error to file
|	pipe output to named command (filter)
|&	include standard error in pipe

Using Backquotes

Another way to redirect output is with backquotes (`). Whatever is enclosed in backquotes is interpreted first, and the resulting output becomes an argument of the command.

One example is the use of the command `ls -a` to list all files, even hidden ones, in a directory. Suppose you want to empty a given directory. Simply typing

```
rm *
```

won't do the trick, because it will miss the hidden files (i.e, those starting with a period). However, this command will work:

```
rm `ls -a`
```
(note backquotes)

This command means "remove all files listed by `ls -a`." Since `ls -a` lists *all* the files in a directory, this command removes all files there.

Backquotes are very useful when you perform operations on the same set of files over and over. To see how this works, create a file that contains the *names* of each of these files:

```
venus% cat > listfile
foreword
preface
introduction
conclusion
venus%
```

Now the output of the construction

```
`cat listfile`
```

is a list of the files to work on, so that

```
cp `cat listfile` /tmp
```

copies all those files (and only those files) to the directory `/tmp`, whereas

```
rm `cat listfile`
```

removes only those files, and so on.

cat *vs.* more

You may have noticed by now that cat and more both display the contents of files (or the standard output). But more breaks the display up by screenful. Therefore, it's better to use more for displays going straight to the screen, and cat for directing output to files or pipelines:

```
venus% cat file | more
(file is displayed here by the screenful)
```

7.7. Command Line Editing

Using the history Command

The C shell keeps a list of previous commands that you have typed in. This history mechanism provides you with the means to select and modify any word from any event in the history list. The history variable determines the length of this list.

You can add this command to your .cshrc file if it isn't already there. For more information on this file and how to change it, see *SunOS User's Guide: Customizing Your Environment.*

To set or change this variable, use a command of the form:

```
set history=n
```

where *n* is the number of commands to remember. (40 is a good number. Setting the history variable to a very high number can degrade performance of the shell.)

To see the list of previous command lines, type history after the prompt. Your history list will be similar to the one below:

```
venus% history
    1  ls
    2  cd
    3  date
    4  history
```

Command Substitution and Repetition

When you want to repeat the previous command line but at the same time change part of it, you will find *command substitution* useful. You can easily substitute one string for another, using two carets (^) to define the operation. This is the syntax:

```
^old^new
```

In the following example, the incorrectly typed command mdkir is corrected to read mkdir. Note that in correcting the command, you also cause it to be performed.

```
venus% mdkir animals
mdkir: Command not found
venus% ^dk^kd
mkdir animals
venus%
```

Adding `^:p` to the substitution line makes the system wait for confirmation before reentering the command. This allows you to reedit your correction, if necessary.

```
venus% mdkir animals
mdkir: Command not found
venus% ^dk^kd^:p
mkdir animals
venus%
```

The command has been edited, printed to the screen, and stored in the history list, but it has not yet been performed. Type `!!` to give the command:

```
venus% mdkir animals
mdkir: Command not found
venus% ^dk^kd^:p
mkdir animals
venus% !!
mkdir animals
venus%
```

Typing `!!` anywhere on the current command line repeats the entire previous command line. You can repeat just the last word of the previous command line by typing `!$` on the current command line, as in this example:

```
venus% cat mammals
chimp
dog
zebra
venus% lpr !$
lpr mammals
venus%
```

You can also use command substitution to streamline repetitive tasks. The next example shows how to use `mkdir` and command substitution to make a series of directories without retyping the whole command line:

```
venus% mkdir plants
venus% ^plants^fungi
mkdir fungi
venus% ^fungi^protistans
mkdir protistans
venus% ^protistans^monerans
mkdir monerans
venus% ls
fungi   plants
monerans    protistans
venus%
```

With longer, more complicated commands, substitution can save a lot of typing (and mistyping!).

For more on command substitution, see *SunOS User's Guide: Doing More.*

Changing Prompt To Display History Number

If you want your prompt to display the history number as well as the hostname, you can use a text editor to change the line in your `.cshrc` file that reads

```
set prompt="`hostname`"
```

to read

```
set prompt="`hostname`_\!%"
```

For more information on changing `.cshrc`, see *SunOS User's Guide: Customizing Your Environment.*

To make your changes take immediate effect, type

```
venus% source ~/.cshrc
venus%
```

Your prompt now looks like this:

```
venus_38%
```

When you start new windows, they will also display the new prompt.

7.8. Processes, PIDs, and Daemons

After each command is interpreted by the C shell, an independent *process*, with a unique process identification number (PID), is created to perform it.

The system uses the PID to track the current status of each process.

What Commands are Running Now? (ps)

To see what processes you have running, use the `ps` command. In addition to showing the PID for each process you own (created as a result of a command you typed in), `ps` also shows you the terminal from which it was started, its current status (or *state*), the cpu time it has used so far, and the command it is performing.

```
venus% ps
  PID TT STAT  TIME COMMAND
 2649 co IW    0:23 sunview
 2650 p0 IW    1:12 shelltool -C
 2651 p0 IW    0:06 -bin/csh (csh)
 6006 p1 R     0:02 ps
 2655 p2 S    34:32 shelltool
 2659 p2 IW    0:50 -bin/csh (csh)
 6000 p2 R     0:05 vi proc
```

The table below should help decipher the display:

Table 7-2 *Information Displayed by* `ps`

Column	*Symbol*	*Meaning*
`PID`		process ID number
`TT`		terminal:
	`co`	`/dev/console`
	mn	`/dev/tty`*mn*
`STAT`		state of the process:
	`R`	runnable (running)
	`T`	stopped
	`P`	paging
	`D`	waiting on disk
	`S`	sleeping (less than 20 seconds)
	`I`	idle (more than 20 seconds)
	`Z`	terminated, control passing to parent
	`W`	swapped out
	`>`	exceeded soft memory limit
	`N`	priority was reduced
	`<`	priority was raised
`TIME`		processing time (so far)
`COMMAND`		command being performed

Of the various states in the `STAT` column, `IW` and `ID` can indicate that a process is in trouble. If you find a process in one of these states, and if in 5 minutes or so it is still in that state, it is probably a good idea to terminate it and run the command again (checking to be sure that the command line makes sense and is typed in correctly).

Terminating a Process With `kill`

`kill` provides you with a direct way to stop commands that you no longer want, even from a shell running on another terminal or from another window. This is particularly useful when you make a mistake typing in a command that takes a long time to run, such as `troff`.

To terminate a process, type `ps` to find out the PID. When you see which process or processes to terminate, type in `kill` followed by the PID s for those processes.

A faster way to get the right PID is to pipe `ps` output through `grep`:

`ps | grep` *command-name*

`grep` is explained in Section 7.11.

```
venus% ps
  PID TT STAT  TIME COMMAND
 1291 co I     0:12 iconedit -Wp 46 90 -Ws 428 276 -WP 176 24 -Wi
 1282 p0 S     0:11 -bin/csh (csh)
 3250 p0 R     0:00 ps
 1286 p1 I     0:05 -bin/csh (csh)
 3248 p1 S     0:05 vi commands
 1287 p2 I     0:40 cmdtool -Wp 496 719 -Ws 645 172 -WP 1088 256
venus% kill 1282
[1]   Terminated      -bin/csh (csh)
venus%
```

Use

```
kill -1 PID#
```

to forcefully terminate a process. If that doesn't work, use

```
kill -9 PID#
```

See Section 7.10 for how to kill commands in the background.

7.9. Aliases

Aliases allow you to substitute a short command for a long one—or a single command for a series of commands. By using aliases, you can save a lot of typing when performing actions you do frequently.

To declare an alias, use the following format:

```
alias name-of-alias 'alias-commands'
```

where *name-of-alias* is the name of the alias; and *alias-commands* is the action (or collection of actions) that this alias performs.

SunOS User's Guide: Customizing Your Environment contains a number of useful aliases you may want to use.

One way to use aliases is to invent an alias that is an abbreviation of a longer command. In the example below, the `h` alias is a quick way to perform the `history` command:

```
venus% alias h 'history'   (create h alias)
venus% h
1 cd
2 ps
3 alias h history
4 h
venus%
```

Another way to use aliases is to create alternate forms of existing commands. One example is the `rm` command. Normally, `rm` removes files without asking for confirmation. With the `-i` option, however, `rm` queries you about deleting files. The example below shows you how to create an alias, `del`, that is equivalent to `rm -i`:

```
venus% alias del 'rm -i'    (create del alias)
venus% del file1
del: remove file1? y
venus%
```

You do not have to create an alias with a new name; aliases can have the name of existing commands. In the next example we create an aliased version of `rm` that is the same as `del`:

```
venus% alias rm 'rm -i'    (create rm alias)
venus% rm file2
rm: remove file2? y
venus%
```

Argument Designators

You can use *argument designators* in alias definitions to create aliases for complicated commands and pipelines. When you use the alias as a command, the argument designator is replaced by command line arguments.

One argument designator is `\!*`. When you use the alias, `\!*` is replaced by all the arguments that follow the alias. An example is an alias for `cd` that not only changes you to the directory you specify, it uses `pwd` to print out that new directory.

Note that the example below uses a semicolon (`;`) to combine two commands into one alias:

`..` is an abbreviation for the parent directory of the current directory.

```
venus% alias cd 'cd \!* ; pwd'    (create cd alias)
venus% cd /etc/uucp    (/etc/uucp replaces \!*)
/etc/uucp                         (cd displays new location)
venus% cd ..    (go to parent directory; .. replaces \!*)
/etc                              (cd displays new location)
venus%
```

Besides `\!*`, you can specify which arguments to replace: `\!1` replaces the first argument to the alias, `\!2` the second, and so forth.

Making Aliases Global and Permanent

Aliases are only valid for the window you type them in; if that window goes away, so do its aliases. To avoid having to type in all your aliases in all your windows, put them into the file `.cshrc`, located in your home directory. Then type

```
source ~/.cshrc
```

in each window you want them in. When you start a new window (or log in), the new aliases will be valid automatically.

Seeing Current Aliases

To see what aliases you have, just type `alias`. To see a particular alias, type `alias` followed by the command you want to see:

```
venus% alias rm    (check rm)
rm -i
venus% alias       (check all aliases)
a       alias
h       history
j       jobs -l
ls      ls -F
mv      mv -i
rm      rm -i
venus%
```

Escaping an Alias

To run the unaliased version of a command, precede the name of that command with a backslash. Here, `rm` is aliased to confirm file deletions, but in its escaped form it removes the file without checking first.

```
venus alias rm         (check rm alias)
rm -i
venus% rm test         (use alias version of rm)
rm: remove test? n
venus% \rm test        (don't use alias version of rm)
venus%
```

The same result can be had by using the command's full pathname:

```
venus% /bin/rm test
venus%
```

Some commands, such as `cd` and `pushd`, are built into the C shell. As such, they cannot be escaped with a backslash. To escape these commands, put the *null string* before the command. The null string is represented by a set of empty double quotes (" "):

The semicolon separates two commands.

```
venus% alias cd 'cd \!* ; pwd'     (create cd alias)
venus% cd
/home/venus/medici
venus% ""cd /usr/lib      (use nonalias cd)
venus%
```

Unaliasing an Alias

To remove an alias, simply use the `unalias` command:

```
venus% alias rm      (check rm alias)
rm -i
venus% unalias rm    (remove it)
venus% alias rm      (check that it's gone)
venus%
```

7.10. Running Commands in the Background

The SunOS operating system is a *multi-tasking* operating system. This means that it can keep track of several users and their commands simultaneously. The system also allows you to run several commands at once by placing them in the *background.*

Running a command in the background means that you can type in other commands while it's running. To run a command in the background, end the command line with an ampersand (`&`):

The `nroff` text formatter is explained in Chapter 6.

```
venus% nroff -ms text.file &
[1] 4001
venus%          (you can type commands while nroff runs)
```

`4001` is a PID number (described in Section 7.8). `[1]` is the *job number.*

To see what jobs are running, use the `jobs` command:

Because each window runs with a different shell, you can't use `jobs` to inquire about stopped jobs in other windows.

```
venus% jobs
[1] + Running           nroff -ms text.file
[2] - Stopped           vi malleable
venus%
```

In the example below there are two stopped jobs. Note the plus sign (+). It indicates that one job is current; the minus sign (–) shows that the other job is waiting.

You can also send programs that are interactive, like `vi` or a game, to the background temporarily. You can stop the job in the middle, go do something else, and then start it up again where you left off. To do so, use Ctrl-Z.

Suppose you're in the middle of editing a file, and you want to leave temporarily:

```
When in the course of human events, it becomes
necessary for one people, in order to secure the
rights of liberty, fraternity, and a decent,
well-aged Gouda cheese, to
Esc
     (leave input mode)
Ctrl-Z         (stop vi)

Stopped

venus%       (now you can type in a command)
```

To restart a command, or bring it to the *foreground*, type

%*n*

where *n* is the job number. If there is only one stopped job, or if you are bringing the current job back to the foreground, you can omit the job number.

To abort a background job, use a command of the form:

kill %*n*

where *n* is the number of the job to kill:

```
venus% kill %1
[1]   Terminated          nroff -ms text.file
```

Exiting With Stopped Jobs

If you try to exit a shell while a job is stopped, you get the warning message:

```
There are stopped jobs.
```

A second `logout` will then log you out (but its a good idea to see what jobs are stopped with `jobs` before you exit).

`bg` and `fg`

The C shell has two built-in commands, `bg` and `fg`, which can be used to put jobs in the background or foreground.

Typing `bg` at a prompt puts the current job in the background. (This has the same effect as if you had ended the original command line with an `&`.) You can specify a job number by typing

bg %*n*

Similarly, typing `fg` brings the current job into the foreground, while another job can be specified by typing

fg %*n*

The at Command

You can take advantage of hours when the system is not heavily used to run large jobs that require a large amount of system time or memory (like formatting large documents with troff or printing out large files; see Chapter 6 for more on this).

First, create a file containing the command line(s) you wish to run later on:

```
venus% cat > atfile:
/usr/bin/troff -ms -t big.doc | lpr -t
Ctrl-D
venus%
```

Then type in at, followed by the time you wish to run the job, and the name of the file containing the command line(s):

```
venus% at 2a atfile
venus%
```

This command tells the system to start formatting and printing the large document at 2:00 a.m.

at accepts a wide variety of syntaxes for specifying the time to run the job; these, for example, are all legitimate:

```
at 1415am Jan 3
at 2:15 pm Jan 3
at now + 1 day
at 5pm Friday
at noon
```

Note that at makes its own copy of the file you want it to execute. If you later change or delete the file, at will not be affected — it will still execute its copy of the original file. Note also that the script should expect no arguments, as none will be passed to it.

There are two other ways you can use at, without creating an "atfile." One is to pipe a command to at:

```
venus% nroff -ms much.too.large.doc | at 1 am
job 937 at Tue Jan  3 01:00:00 1989
venus%
```

The other way is to go into at itself (by typing at), and then typing the command at the prompt. End with Ctrl-D:

```
venus% at midnight
at> nroff -ms much.too.large.doc
at> Ctrl-D
venus%
```

There are two files, `at.allow` and `at.deny`, which regulate who can use the `at` command.

`atq` and `atrm`

`atq` tells you what jobs `at` is waiting to perform.

```
venus% atq
Rank     Execution Date     Owner      Job #    Queue   Job Name
 1st    Jan  3, 1989  0:00  barty        936        a   make
 2nd    Jan  3, 1989  1:00  barty        937        a   lpr -n
```

`atrm` removes jobs from the `at` queue.

- To remove a specific job, type

  `atrm` *`job#`*

  You get the job number by using `atq`.

```
    venus% atrm 936
    936: removed
    venus%
```

- To remove all your queued jobs, type either `atrm -a` or `atrm` followed by your username. (The superuser can remove all queued jobs with `at -a`.)
- To be cautious, use the `-i` option; `atrm` will then ask you to confirm all deletions.

`batch`

`batch` is similar to `at` except that it sends the jobs off immediately to be executed, but waits until the system load level is low before actually *running* them. For more on `at` and `batch`, see the *SunOS Reference Manual* or type **`man at`**.

7.11. Searching for Patterns With `grep`

To search for a particular character string in a specified file, use the `grep` command. The basic syntax of the `grep` command is:

```
grep string file
```

A *string* is one or more characters; a single letter is a string, as is a word or a sentence. Strings may include "white space," punctuation, and invisible (control) characters.

where *string* is the word or phrase you want to find, and *file* is the file to be searched. As shown many times in this chapter, you can also use `grep` with output redirection:

```
command | grep string
```

To find Edgar Allan Poe's telephone extension, type `grep`, all or part of his name, and the file containing the information:

```
venus% grep Poe extensions
Edgar Allan Poe   X72836
venus%
```

Note that more than one line may match the pattern you give:

```
venus% grep Allan extensions
Ethan Allan       X6438
Edgar Allan Poe   X72836
venus% grep Al extensions
Louisa May Alcott X4236
Ethan Allan       X6438
Edgar Allan Poe   X72836
venus%
```

`grep` is case-sensitive; that is, you must match the pattern with respect to upper- and lowercase letters:

```
venus% grep allan extensions    (grep fails)
venus% grep llan extensions
Ethan Allan       X6438
Edgar Allan Poe   X72836
venus%
```

`grep` as a Filter

`grep` is very often used as a filter with other commands. It allows you to winnow out useless information from the output of commands. The example given at the beginning of this chapter illustrates `grep` as a filter; it displays files ending in ".ps" that were created in the month of May:

Pipes and filters are explained in Section 7.6.

```
ls -l *.ps | grep May
```

The first part of this command line,

```
ls -l *.ps
```

produces a list of files:

```
venus% ls -l *.ps
-rw-r--r--  1 elvis        7228 Apr 22 15:07 change.ps
-rw-r--r--  1 elvis        2356 May 22 12:56 clock.ps
-rw-r--r--  1 elvis        1567 Jun 22 12:56 cmdtool.ps
-rw-r--r--  1 elvis       10198 Jun 22 15:07 command.ps
-rw-r--r--  1 elvis        5644 May 22 15:07 buttons.ps
venus%
```

The second part,

```
| grep May
```

pipes that list through `grep`, looking for the pattern `May`:

```
venus% ls -l *.ps | grep May
-rw-r--r--  1 elvis        2356 May 22 12:56 clock.ps
-rw-r--r--  1 elvis        5644 May 22 15:07 buttons.ps
venus%
```

`grep` With Multi-Word Strings

To find a pattern that is more than one word long, enclose the string with single or double quotation marks.

```
venus% grep "Louisa May" people
Louisa May Alcott X4236
venus%
```

`grep` can search for a string in groups of files. When it finds a pattern that matches in more than one file, it prints the name of the file, followed by a colon, before the line matching the pattern:

```
venus% grep ar *
actors:Humphrey Bogart
alaska:Alaska is the largest state in the United States.
wilde:book.  Books are well written or badly written.
venus%
```

Searching for Lines Without a Certain String

To search for all the lines of a file that *don't* contain a certain string, use the `-v` option to `grep`. For example, to find all of the lines in the user `medici`'s home directory files that don't contain the letter `e`:

```
venus% ls
actors        alaska     hinterland   tutors        wilde
venus% grep -v e *
actors:Mon Mar 14 10:00 PST 1936
wilde:That is all.
venus%
```

More on `grep`

Some characters with special meaning to `grep` also have special meaning to the system and need to be quoted or escaped. So, whenever you use a `grep` regular expression on the command line, surround it with quotes, or escape such characters as `&` `!` `.` `*` `$` `?` and `\` with a backslash.

A caret (`^`) indicates the beginning of the line. So the command

```
grep '^b' list
```

finds any line in `list` starting with "b."

A dollar-sign (`$`) indicates the end of the line. The command

```
grep 'b$' list
```

displays any line in which "b" is the last character on the line. Therefore, the command

```
grep '^b$' list
```

displays any line in `list` where "b" is the *only* character on the line.

Within a regular expression, dot (`.`) finds any single character. So the command

```
grep '^.b' list
```

finds all lines in which `b` is the second character. The command

```
grep 'an.' list
```

would match any three characters with "an" as the first two, including "any," "and," "management," and "plan " (because spaces count, too).

When an asterisk follows a character, `grep` interprets it as "zero or more instances of that character." When the asterisk follows a regular expression, `grep` interprets it as "zero or more instances of characters matching the pattern."

Because it includes zero occurrences, usage of the asterisk is a little non-intuitive. Suppose you want to find all words with the letters "qu" in them. Saying

```
grep 'qu*' list
```

will work as expected. But if you wanted to find all words containing the letter "n," you'd have to type

```
grep 'nn*' list
```

and if you wanted to find all words containing the pattern "nn," you'd have to use

```
grep 'nnn*' list
```

You may want to try this to see what happens otherwise.

To match zero or more occurrences of *any* character, use

```
.*
```

Searching for Metacharacters

Suppose you want to find lines in the text that have a dollar sign ($) in them. Preceding the dollar sign in the regular expression with a backslash (\) tells `grep` to ignore (*escape*) its special meaning. This is true for the other metacharacters as well — `& ! . * ?` and \ itself.

For example, the expression

```
^\.
```

matches lines starting with a period, and is especially useful when searching for `nroff` or `troff` formatting requests (which begin with a period). See Chapter 6, Formatting and Printing Documents.

Table 7-3 `grep` *Search Pattern Elements*

Character	Matches:
`^`	The beginning of a text line.
`$`	The end of a text line.
`.`	Any single character
`[...]`	Any single character in the bracketed list or range.
`[^...]`	Any character not in the list or range.
`*`	Zero or more occurrences of the *preceding character* or *regular expression.*
`.*`	Zero or more occurrences of any single character.
`\`	Escapes special meaning of next character.

Note that these search characters may also be used in `vi` text editor searches. See Chapter 5.

Single *vs.* Double Quotes on Command Lines

As shown earlier, use quotation marks to surround text that you want to be interpreted as one word. For example, `grep` searches all files for the phrase "roger, good buddy":

```
venus% grep "roger, good buddy" *
```

Single quotation marks (`'`) also group multi-word phrases into single units. Single quotation marks also make sure that certain characters, such as `$`, are interpreted literally. (The history metacharacter `!` is always interpreted as such, even inside quotation marks, unless you escape it with a backslash.) In any case, it is a good idea to escape characters such as `&`, `!`, `$`, `?`, `.`, `;`, and \ when you want them taken as ordinary typographical characters.

For example, if you type

```
grep $ list
```

you'll see *all* the lines in `list`; if you type

```
grep '$' list
```

you'll see only those lines with the "$" character in them.

8 Using the Network

Networks provide you with the opportunity to use other machines while logged in on your own machine. You can log in to other machines or you can execute commands without logging in to other machines.

In this chapter, you'll learn

- Network concepts;
- How to log in to other machines from yours;
- How to issue commands to remote machines; and
- How to request status information on other machines.

8.1. What Is a Network?

A *network* is a connection between machines, allowing them to transmit information to one another. Networks are often referred to as being *local area networks* (LANs), which range over a small area, generally less than a few thousand feet; *wide area networks* (WANs), which can span thousands of miles; or *campus area networks* (CANs), which are intermediate in size.

Networks may themselves be groups of networks; such a super-network is called an *internetwork*. For example, you may be part of a network of machines on your building's floor and part of an internetwork connecting your local network with similar networks across the country. As the difference between a network and an internetwork is generally invisible to the user, we will use the term "network" to refer to both networks and internetworks.

Machines participating in a network communicate using a network *protocol*, or shared network language, to transmit the appropriate information to the right place. An *internetwork protocol* — sometimes referred to as a *gateway* or *relay* — links networks together.

Sharing Files Over a Network: NFS and RFS

If you are using a Sun Workstation, you are not necessarily on a network. You may, for example, bring your machine up *single-user* or you may simply be on a machine that is not connected to any network.

Still, most Sun users are on a network of one sort or another. The *Network File System*, or NFS®, allows users on different machines to share files across a network. NFS is both a communications protocol — a set of rules machines follow for talking with each other — and a collection of software utilizing that protocol. NFS makes use of services provided by a network to allow for transparent transfer and access of files between machines; that is, NFS allows you to treat some files on other machines as though they were on your own machine. NFS was developed by Sun Microsystems and has been adopted by a number of other companies.

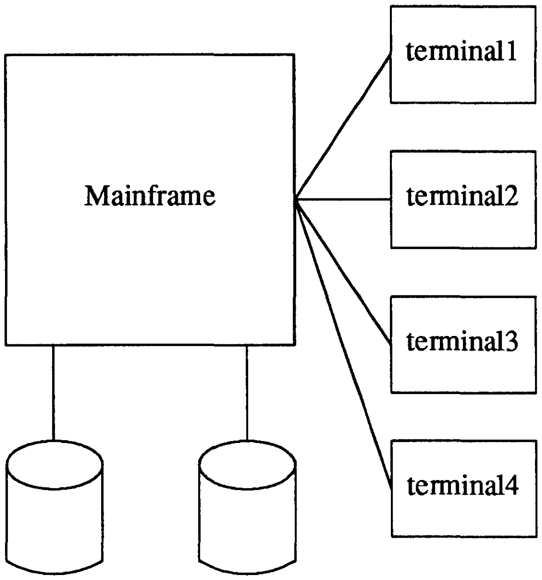
NFS is not limited to machines that run the SunOS operating system; many other systems, from personal computers to super-computers, can use it to share files across a network.

The *Remote File System*, or RFS, is similar to NFS but was developed independently by AT&T.[3] It performs the same function as NFS. Most sites running SunOS will be using NFS; check with your system administrator to see which you are running. *SunOS User's Guide: Customizing Your Environment* explains how to use RFS instead of NFS.

NFS and RFS don't exist on the network — a network is just a lifeless collection of wires, cables, phones, and other communications devices. Rather, each machine utilizing NFS and RFS runs its own set of the NFS and RFS daemons. (A *daemon* — sometimes known as a *server* — is a program that runs on your machine, doing various specific housekeeping chores. A printer daemon might handle the queueing up and printing of files, while a mail daemon takes care of sending messages back and forth between users. Most daemons are invisible to the user.)

Traditionally, a multi-user system had a single processor (or set of processors) and disks serving a number of users:

Figure 8-1 *A Non-Network Environment*

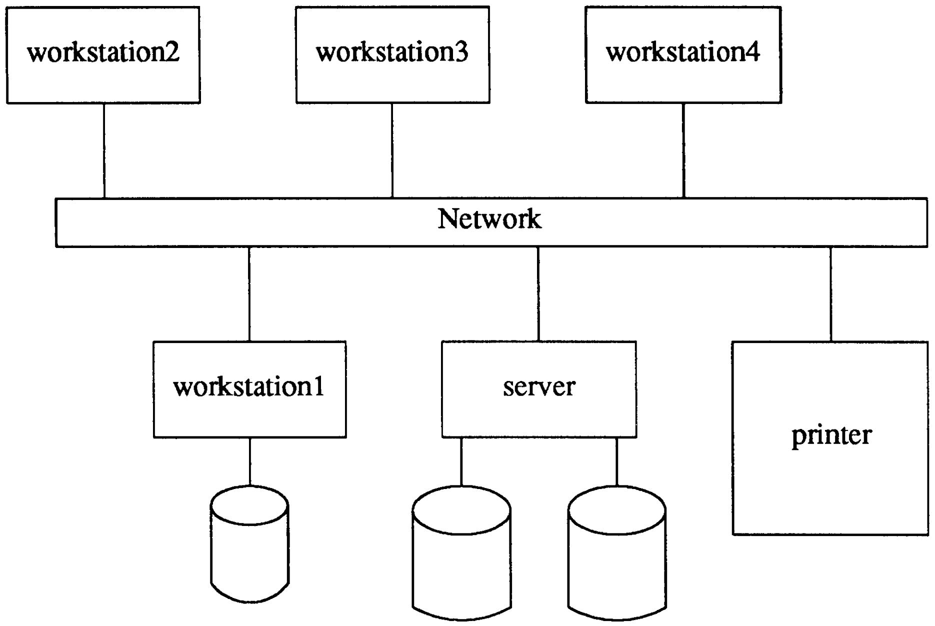


The problem with the set-up in Figure 8-1 is that all the users are competing for the same processor. With a network of workstations, however, each user has his

[3] RFS and "Remote File System" are trademarks of AT&T.

or her own central processor and can access files located on other machines. NFS allows files to be scattered among the various machines on the network, so to the user the whole network appears somewhat like one big computer:

Figure 8-2 *A Typical Network Set-Up*



Using NFS and RFS, you can get access to other machine's files by *mounting* file systems on those machines. *SunOS User's Guide: Customizing Your Environment* gives an introduction to mounting remote file systems under NFS and RFS; for a fuller discussion, see the *System and Network Administration* manual.

Types of Networks

Some common networks you may encounter include:

Local Area Networks
As mentioned earlier, small networks are sometimes called local area networks (LANs). In local area networks, machines are connected by cables. Two types of LAN networks common to Sun machines are based on Ethernet and FDDI technologies; you may, for example, hear references to "sending something over the Ethernet." Such networks allow real-time communication between machines, so you can log in directly to another machine and run programs as though it were in front of you. You can also copy files to and from remote machines on a LAN.

ARPANET-Based Networks
In 1969 the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense sponsored the development of a broadscale communication protocol and network. Today, the descendant of that effort is called the Internet, which is composed of a number of interconnected networks that use the Internet protocol (IP).

Users on machines that attach directly to the Internet, by cables or microwave connections, can log in ''real-time'' to other machines on the network.

UUCP Network

The UUCP ("Unix-to-Unix Copy Program") network is not a permanent system of hardware connecting machines but rather a program that allows machines to use telephones to transmit data. Thus UUCP creates a sort of ad hoc network each time the program is invoked. (People disagree on whether it should be considered a network at all.) You can use UUCP to communicate with machines across the United States and throughout the world.

An ad hoc UUCP network is not a real-time network like a LAN network; in other words, you can't log in to another machine using UUCP. However, you *can* send information, like mail messages, to other machines and receive answers back from them.

Because UUCP is not a real-time network, most of the information in this chapter— such as logging in to remote machines, copying from other machines, and issuing commands to other machines— does not apply to it. For information about how to send and receive mail messages on the UUCP network, see the mail chapters in this manual.

People using machines on a UUCP network sometimes can send mail messages over ARPANET-based networks by utilizing a *mail gateway*, a machine that transfers data from one network to another.[4]

The Network Information Service (NIS)

Because networks can get very large, doing system administration on them can get quite complicated: one needs to keep track of who belongs to a network, who has permissions to access which filesystems, and so on. For this reason, Sun provides the Network Information Service, or NIS. NIS maintains certain files with information about the machines which belong to the network and their users, and it provides users all over the network with the ability to look up the information contained in those files. By default there are a number of databases in NIS, but a system administrator may delete or add databases as desired. These default databases include `hosts`, which has machine names along with their addresses on the network; `passwd`, which, like your local `/etc/passwd` file, contains password and other information associated with usernames; `group`, which holds the various groups that users have divided themselves into; and `aliases`, which contains mail aliases that the network recognizes. Not all systems run NIS — ask your system administrator if yours does.

NIS is a *distributed* database; that means that copies of its data files are kept in various places throughout the network, for faster access by users. (There is one central, master copy, and changes to it are propagated down to the other "slave" copies.) NIS revolves around two daemons: `ypserv`, which is the lookup function on the machine that has a copy of NIS, and `ypbind`, which runs on your machine and determines where to obtain NIS. In order to use NIS, these two

[4] For information about how to send and receive mail messages on ARPANET-based networks, see Chapter 13.

daemons must be running. To find out if your machine is using NIS, use the `ps` and `grep` commands; you should see something like this:

```
venus% ps -aux | grep ypbind
root    27  0.0  1.1   66   34 ?  I    0:15 /usr/etc/ypbind
venus%
```

(You can use `ps -aux` to check for the presence of any daemon, including the NFS daemons.)

You may never need to access the NIS files; generally only system administrators ever tinker with them. However, you may occasionally want to read these files. Some commands to help you include the following:

`ypcat`
: This displays the NIS data file you name. You can pipe it to the `grep` command to find the entries you want.

`ypmatch`
: `ypmatch` looks up an entry, or "key," in the NIS file you specify.

`ypwhich`
: This command tells you which server supplies NIS services for your machine.

`yppasswd`
: You use `yppasswd` to change your network, as opposed to your machine's, password. (Often they're the same.)

For further information on these commands, type `man` followed by the command name. For a fuller treatment of NIS, type **`man ypfiles`** or consult the *System and Network Administration* manual.

8.2. Remote Login With `rlogin`

`rlogin` logs you in to other UNIX machines on a network.

Logging in to Another Machine With `rlogin`

Type `rlogin` and the *machine name* of the other machine. Should a password prompt appear, type the password for that machine followed by [Return]. If your machine's name is in the other machine's `/etc/hosts.equiv` file, then the other machine trusts your machine name and won't require you to type the password.

```
venus% rlogin jupiter
Password: (Type password)
Last login: Mon Oct 20 00:30:52 from venus
SunOS Release 4.1 (SUN) #9: Sat Nov 16 12:51:59 PST 1989
jupiter% pwd
/home/medici
jupiter% logout
Connection closed.
venus%
```

`rlogin` to a Machine Where You Don't Have a Home Directory

In the example above, user `medici` logged in to `jupiter` at the directory `/home/medici`, as indicated by the `pwd` command. When you log in to a machine where you don't have a home directory, `rlogin` displays a notification that you have no home directory on that machine and logs you in to the root directory (`/`) of that machine:

```
venus% rlogin neptune
Password:
No directory! Logging in with home=/
Last login: Mon Nov 25 16:58:57 from venus
SunOS Release 4.1 (SUN) #9: Sat Nov 16 12:51:59 PST 1989
neptune% pwd
/
neptune% logout
Connection closed.
venus%
```

`rlogin` to a Machine as Someone Else

`rlogin` as we've described it above allows you to log in to another machine, but under your own username. At times you may want to log in as someone else. One example of this would be when you're off working on someone else's machine (and using their username) and you want to log in to your own machine as yourself. The `-l` option to `rlogin` allows you to do this. The command syntax is:

```
rlogin machine-name -l username
```

For example, here's how user `medici` on machine `venus` logs in to machine `aphrodite` as `cosimo`:

```
venus% rlogin aphrodite -l cosimo
Password:
Last login: Tue Nov 26 00:02:00 from venus
SunOS Release 4.1 (SUN) #9: Sat Nov 16 12:51:59 PST 1989
aphrodite% pwd
/home/cosimo
aphrodite% logout
Connection closed.
venus%
```

Note that when you log in as someone else, you go to that person's home directory.

`rlogin` to a Non-Existent Machine

If you attempt to log in to a machine whose name isn't known to your machine, say the machine `andromeda`, `rlogin` searches unsuccessfully through the hosts database for that machine, then displays the following notification:

```
venus% rlogin andromeda
andromeda: unknown host
venus%
```

If you see error messages that you don't understand, try looking at Table 8-1 , which contains solutions to common networking problems.

Aborting an `rlogin` Connection

Usually you abort an `rlogin` connection only when you can't terminate the connection using `logout` at the end of the work session.

To abort an `rlogin` connection, type a tilde character followed by a period character (`~.`) at the beginning of a line. The login connection to the other machine aborts, and you find yourself back at your original machine.

When you log in to a series of machines, accessing each machine through another machine, and you use `~.` to abort the connection to any of the machines in the series, you return to the machine you started from:

```
venus% rlogin comet
Last login: Thu Nov 21 05:04:03 from venus
SunOS Release 4.1 (SUN) #9: Sat Nov 16 12:51:59 PST 1989
comet% ~. (Sometimes ~ doesn't echo.)
Closed connection.
venus%
```

To disconnect to an intermediate `rlogin`, use two tildes (`~~.`). For example:

```
venus% rlogin comet
comet% rlogin jupiter
jupiter% ~~. (Sometimes ~~ doesn't echo.)
comet%
```

Suspending an `rlogin` Connection

When you want to *suspend* an `rlogin` connection, so that you can return to it later, type the tilde character (~) followed by [Ctrl-Z]. The `rlogin` connection becomes a *stopped process*, and you are put back into the machine you logged in from.

To reactivate the connection, type `fg`. Alternatively, type `%` followed by the *job number* of the stopped process (default job number for `%` is the job you most recently stopped or put in the background).

```
venus% rlogin animation
Last login: Thu Nov 21 07:07:07 from venus
SunOS Release 4.1 (SUN) #9: Sat Nov 16 13:21:24 PST 1989
animation% ~^Z    (Sometimes ^Z doesn't echo on the screen.)

Stopped
venus% pwd
/home/medici
venus% %
rlogin animation

animation% logout
Connection closed.
venus%
```

As is the case with aborting `rlogin` with `~~.`, using *two* tildes and a [Ctrl-Z] will suspend you to an intermediate `rlogin`.

Verifying Your Identity With `who am i`

When logged in to a series of remote machines, perhaps under a variety of login names, you may need to verify just who and where you are (relative to your computer, that is). The command

```
who am i
```

displays the machine you're logged into, along with your current identity. (It also gives you the name of your terminal or workstation and the time.)

```
venus% rlogin tripod -l billyboy
Password:
Last login: Tue Nov 26 00:02:00 from portnoy
SunOS Release 4.1 (SUN) #9: Tue Nov 17 21:10:51 PST 1989
tripod% who am i
tripod!billyboy    tty3    Nov 20 20:07
tripod% logout
Connection closed.
venus% who am i
venus!medici    tty3    Nov 20 20:08
venus%
```

For further information about `rlogin`, see its man page, online or in the *SunOS Reference Manual.*

8.3. Who's Logged in to This Machine? `users`, `who`, and `w`

When you want to find out who's logged on to your machine, you can use one of three commands: `users`, `who`, and `w`.

The `users` command displays, in alphabetical order, the username of each person logged in on your machine:

```
venus% users
medici rimbaud
venus%
```

The `who` command provides more information than `users` does. For each terminal running on your machine, `who` displays the username, the terminal name, and the date and time the user created the terminal *process*:

In the SunOS operating system, *processes* execute commands. The process that supports a terminal may run without any actual piece of hardware — what we usually think of as a terminal — associated with it; each window on a Sun workstation counts as a separate terminal.

```
venus% who
medici   console Apr 1 8:50
medici   ttyp0   Apr 1 8:51
medici   ttyp1   Apr 1 8:51
rimbaud  ttyp2   Apr 1 9:36   (verlaine)
venus%
```

When a user has logged in to your machine from another machine, the name of that machine appears enclosed within parentheses after the rest of the information `who` displays about them.

The `w` command gives yet more information. First, `w` displays system information, including the current time, how long since the last *reboot* of your machine, the number of terminals running on the machine, and system load information.

Rebooting a machine is, essentially, starting its software up. For example, you reboot when you turn the power on. You also reboot after your machine "crashes."

For each terminal running on your machine, `w` displays the username, the terminal name, the time of terminal login, other system information, and what program that process is running:

The line in the figure starting with `-Ws` wrapped around, continuing from the end of the previous line.

```
venus% w
  9:43am  up 11:11, 4 users,  load average: 0.76, 0.45, 0.27
User     tty       login@  idle   JCPU   PCPU  what
medici   console   8:50am  4.02   4:40   3:59  clocktool -Wp 120 120
 -Ws 122 55
medici   ttyp0     8:51am     2   5:34   1:14  vi sculptor.list
medici   ttyp1     8:51am 94:14     15     15  date
rimbaud  ttyp2     9:36am     1      5      5  -csh
venus%
```

For more information on `users`, `who`, and `w`, see the appropriate man page, online or in the *SunOS Reference Manual*.

8.4. Status Information: `ping`, `rup`, and `perfmeter`

`ping`, `rup`, and `perfmeter` provide *status information* for other machines.

Dead or Alive: `ping`

`ping` checks to see if another machine is up and running. `ping` is useful because it's a quick way to determine the status of another machine before you try to access it or if you're having trouble accessing it. The `ping` command takes the form

```
ping host timeout
```

where *host* is the machine you're testing. *timeout* is an optional amount of time you want `ping` to try for. If you don't include *timeout*, `ping` will try for a maximum of 20 seconds.

```
venus% ping pong
pong is alive
venus%
```

Remote Uptime: `rup`

To find out the length of time a system has been "up and running" and to view its load average, type `rup` followed by the *machine name* of the desired machine:

```
venus% rup pluto
       pluto    up  1 day,   5:47,    load average: 3.64, 3.53, 2.80
venus%
```

`rup` stands for remote **up**time, a version of the `uptime` command that runs `uptime` on other machines.

To get the same set of information for all of the machines that are "up and running" and on your local network, type `rup` without any arguments:

```
venus% rup
     mercury    up  1 day,  23:31,    load average: 0.04, 0.00, 0.00
       venus    up  8 days, 21:27,    load average: 0.26, 0.13, 0.02
       earth    up  3 days, 20:44,    load average: 1.07, 0.90, 0.54
        mars    up          36 mins,  load average: 0.01, 0.00, 0.00
     jupiter    up          20:30,    load average: 0.52, 0.20, 0.05
      saturn    up  8 days, 18:29,    load average: 1.27, 1.12, 1.00
     neptune    up  2 days,  2:38,    load average: 2.27, 1.73, 1.18
      uranus    up  1 day,   3:19,    load average: 0.64, 0.90, 1.04
       pluto    up  1 day,   5:47,    load average: 3.64, 3.53, 2.80
venus%
```

To find out more, see the `rup` man page, online or in the *SunOS Reference Manual.*

Remote User Information With `rusers`

`rusers` tells you who's logged on to other machines on your network. The command `rusers` by itself shows each machine on the network, followed by the users on that machine:

rusers stands for remote users, a version of the users command that runs users on other machines.

```
venus% rusers
homeplate      billyb
absalom        john
babylon        manchild
garden         russell     cousy     havilcek
texas          houston
venus%
```

Note that machine garden has three different users on it.

If you wanted information on just one machine, you would type rusers followed by the machine name:

```
venus% rusers garden
garden         russell     cousy     havilcek
venus%
```

For more detailed information, there's the -l option, which gives the username, the machine and terminal names, the time the user logged on, how long the users's been idle (if more than one minute), and the name of a machine that the user rlogged in from, if any:

```
venus% rusers -l garden
russell     garden:ttyd8      Nov 18 09:00    7:24
cousy       garden:console    Nov 18 16:13
havilcek    garden:ttyp0      Nov 18 11:43      21 (quirk)
venus%
```

The -l option can also be used when no machine name is given. For more information, see the rusers man page, online or in the *SunOS Reference Manual.*

The finger Command

Unlike commands such as ping or rusers, finger doesn't give you information about other machines: it tells you only about other users. In fact, finger is so user-oriented that it accepts people's real names, as well as their usernames, as arguments.

Here's what finger tells you:

- the user's login name
- the user's real name
- the user's home directory and login shell
- the last time the user logged in
- the last time the user received mail, and the last time it was read
- the name of the user's terminals and how long they've been idle

Here is a slightly simplified example of a two typical finger requests:

```
venus% finger moby@sea
[sea]
Login name: moby                    In real life: Ishmael Wong
Directory: /home/shipwreck/moby        Shell: /bin/csh
On since Nov 14 06:33:41 on console   4 days 14 hours Idle Time
New mail received Wed Nov 18 20:34:02 1987;
  unread since Wed Nov 18 16:20:24 1987
venus% finger Henry Stamper
Login name: hank                In real life: Henry Stamper, Jr
Directory: /home/oregon/hank             Shell: /bin/csh
Last login Wed Oct 21 16:16 on ttyp0 from cairo
No unread mail
```

`finger` is useful for making sure that the user you're looking for is still alive and kicking (computer-wise, anyway). `finger` has various options and further capacities; to learn about them, see the `finger` man page online or in the *SunOS Reference Manual.*

8.5. Remote Command Execution With `rsh`

`rsh` stands for remote shell, or an interpreter capable of executing commands on another machine.

`rsh` allows you to execute a single command on another machine without having to log in formally . It can save time when you know you only want to do one thing on the remote machine.

To execute a command on another machine, type

rsh *machinename command*

For example, suppose you want to see the contents of the directory `/home/fresno/crops` on the machine `fresno`:

```
venus%  rsh fresno ls /home/fresno/crops
corn            kiwi            oranges
grapes          olives          peaches
venus%
```

When you execute a command on another machine using `rsh`, `rsh` doesn't log in: it talks to a daemon that executes the command on the other machine. However, if you have a `.cshrc` file in your home directory *on the other machine*, `rsh` reads it. So `rsh` uses any pertinent aliases that you have defined on the other machine when executing the command.

Like `rlogin` and `rcp`, `rsh` uses the other machine's `/etc/hosts.equiv` and `/etc/passwd` files to determine whether you have unchallenged access privileges.

For more information, see the `rsh` man page online or in the *SunOS Reference Manual.*

Table 8-1 *Network Error Notifications: Problems and Solutions*

Error Notification	*Problem*
`Address already in use.`	Another server is "listening" on that address.
`Connection refused.`	The other machine is up, but its daemons aren't ready to complete a connection.
`Connection timed out.`	One machine or the other is off, hung (stuck), or down.
`File not found.`	File doesn't exist on other machine.
`Host name for your address unknown.`	Other machine needs your machine name in its `/etc/hosts` file or its NIS map.
`Login incorrect.`	You mistyped a password, or your username isn't in the other machine's `/etc/passwd` file or NIS map.
`Network is unreachable.`	Routing problem — a gateway machine or other network connection is broken.
`No such file or directory`	File or directory on other machine doesn't exist, or you don't have read permission.
`Permission denied.`	Machine name isn't in `/etc/hosts.equiv` file; password required; may need write permission for directory on your machine.
`RPC_PMAP_FAILURE`	Daemon not running properly.
`RPC: Timed out`	Couldn't connect to other machine, it may be down or network overloaded.
`RPC: Unable to send`	System cannot send data.
`RPC: Unable to receive`	System cannot receive data.
`RPC_UNKNOWNHOST`	Machine name doesn't exist on network.
`unknown host`	Other machine name not in `/etc/hosts` file on your machine.
`NFS server not responding still trying`	Server is under heavy load or has crashed.

9 Messages

This chapter describes messages, so that you can communicate with other users more immediately and interactively than by electronic mail. See Chapters 10 and 11 for more on electronic mail.

There are three kinds of electronic messages:

- Interactive messages with `talk` or `write`
- Broadcast messages with `wall`
- System messages from your machine

9.1. Using `talk`

With the `talk` program, you can converse on your screen with someone else who is either using a terminal on your machine or using another machine on your local network.

To start `talk`, type

`talk` *`username@machinename`*

at your command prompt, followed by [Return].

When using SunView, initiate your `talk` message session in the window you want to use for the session. Pick a window that is large enough to contain a fair amount of text.

In this example, user `medici` attempts to contact user `michaelangelo`.

```
venus% talk michaelangelo@david
```

`talk`'s interactive screen appears, and `talk` attempts to connect with the other user's machine. Until `talk` connects to the other machine, it displays the notification:

```
                    [No connection yet]
```

Once connected, `talk` notifies you that it is waiting for the other person to respond:

```
          [Waiting for your party to respond]
```

talk "rings" the other person again and again, printing a message repeatedly on the screen while waiting for a response. If the other person isn't a user or isn't logged in at that time, talk responds with:

```
[Your party is not logged on]
```

But when talk finds the other user, the talk interactive screen displays a line to split itself in half like this:

```
[Ringing your party again]
[Ringing your party again]
[Ringing your party again]





---------------------------------------------------------



```

To facilitate a connection, talk displays a message that includes your username and machine name on the other user's screen. In this example, talk displays the following message on user michaelangelo's screen:

```
Message from Talk_Daemon@venus at 0:01 ...
talk: connection requested by medici@venus
talk: respond with:  talk medici@venus
```

The other user must respond by typing talk followed by the username and machine name of the person who is attempting to talk. In our example, michaelangelo types:

```
talk medici@venus
```

to confirm the talk connection with user medici on machine venus.

If michaelangelo is busy or wants to ignore medici, he refuses to answer medici's request. Eventually medici gives up, and types Ctrl-C to exit from the talk interactive screen.

However, if michaelangelo successfully responds to medici's request, talk establishes a link between the two users.

```
[Connection established]
```

Now, both users can type messages on the screen without interfering with each other. Both users see the messages they've typed on the upper half of their own screens or windows; the other user's messages appear on the lower half of their screens.

```
[Connection established]
I sure am hungry.

How long until the party?

OK. Bye. (Type Ctrl-C to terminate connection.)


Well, you can eat at the party.

Let's go now.
```

When they have finished typing messages, *either* user types Ctrl-C to terminate the `talk` message session.

You can prevent `talk` messages from appearing on your screen by giving the command:

```
venus% mesg n
```

If you're running the SunView window system, you must be superuser (`root`) to run `mesg`. You can also put `mesg n` on a line by itself in your `.login` file; this file is consulted whenever you log in, so messages will be turned off until you take the line out and log in again. *SunOS User's Guide: Doing More* contains information on becoming superuser, and *SunOS User's Guide: Customizing Your Environment* explains the `.login` file.

For more information on `talk`, see the `talk` man page, online or in the *SunOS Reference Manual.*

9.2. Using `write`

`write` differs from `talk` in that `write`:

- doesn't use the entire screen or window
- only reaches users on the same machine or workstation you're sending messages from

One user `writes` a message to the other. Then the other user can in turn `write` a reply.

To `write` a message to someone using a terminal on your machine, type

```
venus% write username
```

followed by [Return].

In this example, user `medici` decides to `write` some messages to user `sappho`. He types in the text of an introductory message on lines following the `write` command line. To send the introductory message text, type [Return].

```
venus% write sappho
Do you want to chat?
(Type
[Return]
to send message text.)
. . .
```

The message appears on the other user's screen almost immediately afterwards.

`sappho` decides to exchange messages with `medici`, so she types `write`, followed by his username, [Return], and her message in reply.

Users may want to establish conventions — like "over" and "over-and-out" — to indicate when they've completed their messages.

```
venus%
Message from venus!medici on ttyp2 at 1:01 ...
Do you want to chat?
write medici
Sure, what's up? (Type [Return] on next line to send message text.)
. . .
```

As you can see, `write` automatically identifies the machine, username, and terminal where the message originated, and the time the message arrived.

The two interlocutors can continue to write messages back and forth, without retyping the `write` command, until they want to stop. Then *both* users must type [Ctrl-D] on a line by itself to terminate the `write` connection.

```
venus% write sappho
Do you want to
chat?
Message from venus!sappho on ttyp3 at 1:02 ...
Sure, what's up?
Oops, I'm late for an
appointment - gotta
run!
^D (Terminate connection.)
venus%
```

`write` displays the end-of-file indicator, `EOF` on the other user's screen (for this example, user `sappho`'s screen) to notify that person that her conversational partner (user `medici`) terminated the connection.

Just as with `talk`, you can prevent `write` messages from appearing on your screen by using `mesg n`.

For more information on `write`, see the `write` man page, online or in the *SunOS Reference Manual.*

9.3. Broadcast Messages: `wall`

Most users sharing a machine don't appreciate people sending spurious messages to everyone on the machine.

When you want to send a message to everyone on your machine at once, use the `wall`, **write to all**, command. Usually, people broadcast messages only to announce that the machine is going down for maintenance, or for other important messages that affect everyone using the machine.

Type `wall` followed by [Return]. Then, type the text of the message, followed by [Ctrl-D] on a line by itself. The message appears on the screen — in the console window — almost immediately after you send it.

```
venus% wall
This machine will go down for maintenance at
noon today. (Type [Ctrl-D] on next line to end text, send message.)
^D
Broadcast Message from venus!medici (ttyp4) at 12:00 ...

This machine will go down for maintenance at
noon today.

venus%
```

The same message appears on the screen, or console window, of anyone else who is logged in to that machine.

For more information on `wall`, see the `wall` man page, online or in the *SunOS Reference Manual.*

9.4. System Messages

System messages are like broadcast messages, only the system generates them automatically to notify you about something that may be important. One common system message is the *message of the day*.

When you log in, you often see two system messages — one about the operating system, the other about new mail — shown here as examples:

```
venus login: medici
Password:
Last login: Fri Oct 31 23:59:59 from console
SunOS Release 4.1 (DIONE_CLIENT) #1: Fri Feb 14 00:00:01 PST 1989

You have mail.
venus%
```

10 Mail Tool

10.1. What Is Mail Tool?

Mail Tool is a SunView-based program that simplifies use of SunOS mail-handling facilities. With Mail Tool you can write, send, and receive electronic mail to and from other system users, even if they are physically distant and even if they're not using a Sun system. Mail Tool is built on top of a SunOS program called `mail`, which is is not window-based. (See Chapter 11 for more on `mail`.)

Mail Tool options and defaults are located in an associated file called `.mailrc`, found in your home directory. You can use the Defaults Editor or a text editor to access and change these. See Appendix B.

More information on Mail Tool can be found by typing **`man mailtool`** at a system prompt.

10.2. Who Can Use Mail Tool?

To run Mail Tool, you must have a bit-mapped screen, like the Sun Workstation. And you must be using the SunView windows system. If your machine is not so equipped, you must use `mail`, instead. If you're not sure whether you have a bit-mapped screen, ask your system administrator. See the *SunView User's Guide* for information on how to run SunView.

10.3. Starting Mail Tool

There are two ways to start Mail Tool: from a command line and from a SunView menu.

Starting From a Command Line

To start Mail Tool from a command line, type `mailtool &` at the prompt:

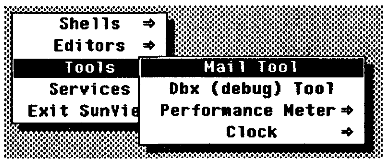
```
venus% mailtool &
[1] 68022
venus%
```

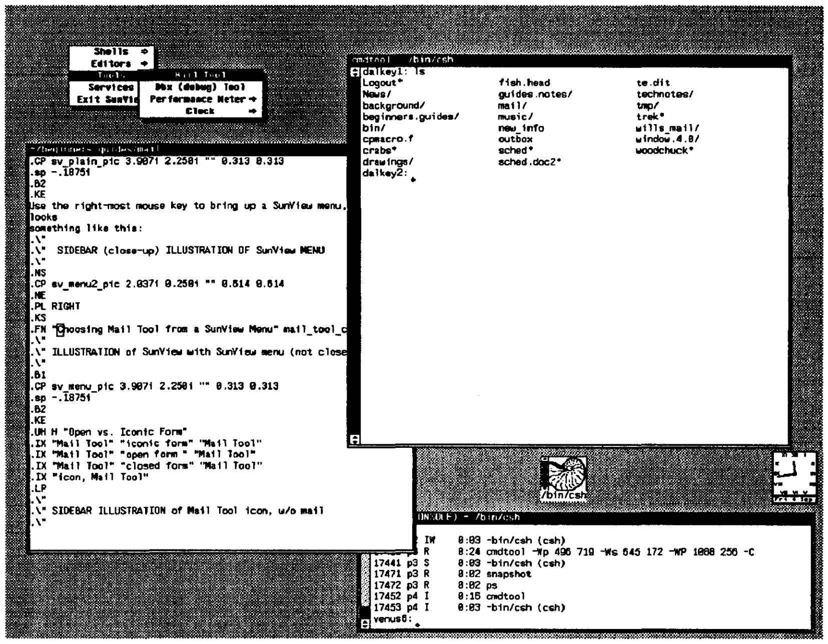
The `&` is optional. It tells Mail Tool to run in the *background*, so you can still do other work in the same window while using Mail Tool.

Starting From the SunView Menu

It's more common to start Mail Tool from the SunView menu.

With the pointer anywhere in the gray area of the screen, hold down the right mouse button to bring up the SunView menu. Choose 'Tools⇒Mail Tool' to select Mail Tool. This selection looks something like this:



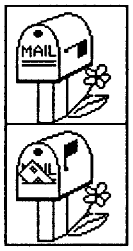


```
.CP sv_plain_pic 3.9071 2.2501 "" 0.313 0.313
.sp -.18751
.B2
.KE
Use the right-most mouse key to bring up a SunView menu,
looks
something like this:
.\"
.\"  SIDEBAR (close-up) ILLUSTRATION OF SunView MENU
.\"
.NS
.CP sv_menu2_pic 2.0371 0.2501 "" 0.514 0.514
.NE
.PL RIGHT
.KS
.FN "Choosing Mail Tool from a SunView Menu" mail_tool_c
.\"
.\" ILLUSTRATION of SunView with SunView menu (not close
.\"
.B1
.CP sv_menu_pic 3.9071 2.2501 "" 0.313 0.313
.sp -.18751
.B2
.KE
.UH H "Open vs. Iconic Form"
.IX "Mail Tool" "iconic form" "Mail Tool"
.IX "Mail Tool" "open form " "Mail Tool"
.IX "Mail Tool" "closed form" "Mail Tool"
.IX "icon, Mail Tool"
.LP
.\"
.\" SIDEBAR ILLUSTRATION of Mail Tool icon, w/o mail
.\"
```

```
cmdtool   /bin/csh
dalkey1: ls
Logout*              fish.head          te.dit
News/                guides.notes/      technotes/
background/          mail/              tmp/
beginners.guides/    music/             trek*
bin/                 new_info           wills_mail/
cpmacro.f            outbox             window.4.0/
crabs*               sched*             woodchuck*
drawings/            sched.doc2*
dalkey2:
```

```
(CONSOLE) - /bin/csh
          IW    0:03 -bin/csh (csh)
          R     0:24 cmdtool -Wp 496 719 -Ws 645 172 -WP 1008 256 -C
17441 p3 S      0:03 -bin/csh (csh)
17471 p3 R      0:02 snapshot
17472 p3 R      0:02 ps
17452 p4 I      0:16 cmdtool
17453 p4 I      0:03 -bin/csh (csh)
venus6:
```

The Mail Tool Icon



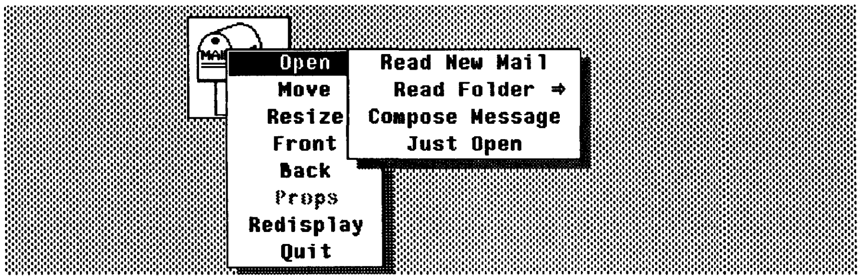
When Mail Tool comes up, it appears as an *icon*. When Mail Tool is displayed in iconic form, it is said to be running *closed*.

10.4. Opening Mail Tool

The icon lets you run Mail Tool without taking up much room on the screen. But to *use* Mail Tool, you have to open it. Use the mouse to move the pointer on the screen into the Mail Tool icon, and then click the left mouse button.

Like most icons, the Mail Tool icon has a menu that allows you to manipulate the application. You can also open Mail Tool with this menu. The 'Open' menu offers the following ways of opening Mail Tool:

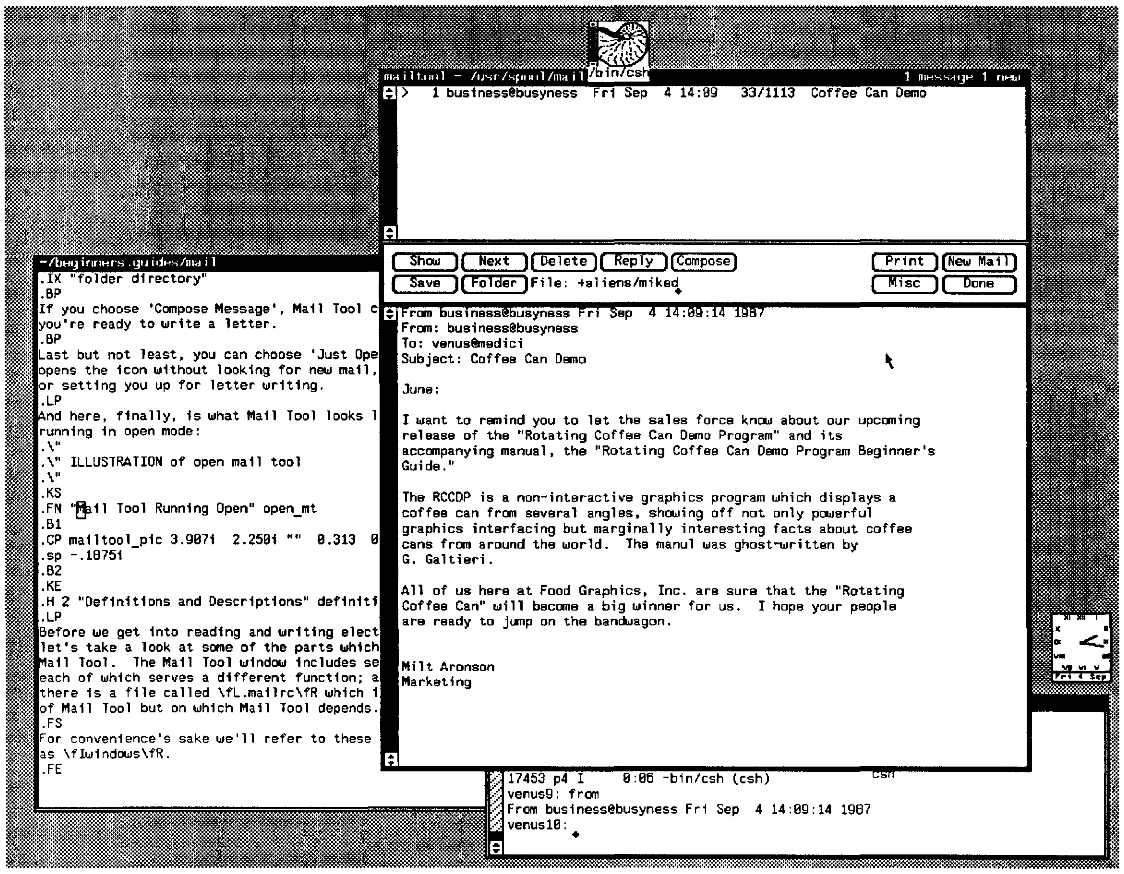
Figure 10-1 *The Iconic Open Menu*



- 'Read New Mail': When you open Mail Tool this way, it automatically checks for new mail.
- 'Read Folder': A folder is a kind of a file that contains letters (it's explained later). Use this item to display your folders and to select the folder you want to look at.
- 'Compose Message': Choosing this item brings Mail Tool up ready for you to write a letter.
- 'Just Open': This choice opens the icon without looking for new mail, reading in a folder, or setting you up for letter writing.

This is what Mail Tool looks like when it's running open:

Figure 10-2 *Mail Tool Running Open*



```
mailtool - /usr/spool/mail                                        1 message 1 new
>   1 business@busyness  Fri Sep  4 14:09   33/1113  Coffee Can Demo
```

```
File: +aliens/miked
```

```
From business@busyness Fri Sep  4 14:09:14 1987
From: business@busyness
To: venus@medici
Subject: Coffee Can Demo

June:

I want to remind you to let the sales force know about our upcoming
release of the "Rotating Coffee Can Demo Program" and its
accompanying manual, the "Rotating Coffee Can Demo Program Beginner's
Guide."

The RCCDP is a non-interactive graphics program which displays a
coffee can from several angles, showing off not only powerful
graphics interfacing but marginally interesting facts about coffee
cans from around the world.  The manul was ghost-written by
G. Galtieri.

All of us here at Food Graphics, Inc. are sure that the "Rotating
Coffee Can" will become a big winner for us.  I hope your people
are ready to jump on the bandwagon.


Milt Aronson
Marketing
```

```
~/beginners.guides/mail
.IX "folder directory"
.BP
If you choose 'Compose Message', Mail Tool c
you're ready to write a letter.
.BP
Last but not least, you can choose 'Just Ope
opens the icon without looking for new mail,
or setting you up for letter writing.
.LP
And here, finally, is what Mail Tool looks l
running in open mode:
.\"
.\" ILLUSTRATION of open mail tool
.\"
.KS
.FN "Mail Tool Running Open" open_mt
.B1
.CP mailtool_pic 3.907i  2.250i ""  0.313  0
.sp -.1075i
.B2
.KE
.H 2 "Definitions and Descriptions" definiti
.LP
Before we get into reading and writing elect
let's take a look at some of the parts which
Mail Tool.  The Mail Tool window includes se
each of which serves a different function; a
there is a file called \fL.mailrc\fR which i
of Mail Tool but on which Mail Tool depends.
.FS
For convenience's sake we'll refer to these
as \fIwindows\fR.
.FE
```

```
17453 p4 I     0:06 -bin/csh (csh)
venus9: from
From business@busyness Fri Sep  4 14:09:14 1987
venus10:
```

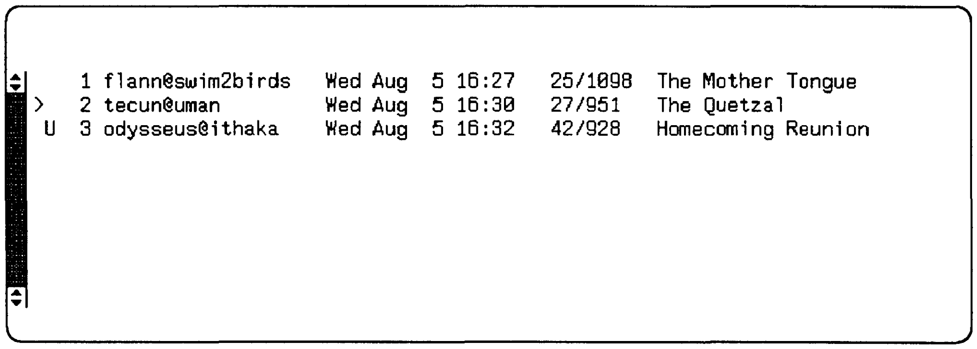
10.5. A Glimpse at Mail Tool Windows

The Mail Tool window includes several subwindows, each of which serves a different function.

The Header List Window

The letters in your mailbox are summarized in the *header list*. Each letter has a one-line entry in the header list called a *header*.

Figure 10-3 *The Header List Window*



```
    1 flann@swim2birds   Wed Aug  5 16:27   25/1098  The Mother Tongue
 >  2 tecun@uman         Wed Aug  5 16:30   27/951   The Quetzal
 U  3 odysseus@ithaka    Wed Aug  5 16:32   42/928   Homecoming Reunion
```

Figure 10-3 shows a header list with three headers. The first letter is from user *flann* on machine *swim2birds*; the second letter is from user *tecun* on machine *uman*, while the third letter is from *odysseus* on machine *ithaka.*

Reading across from left to right, each header displays the following information:

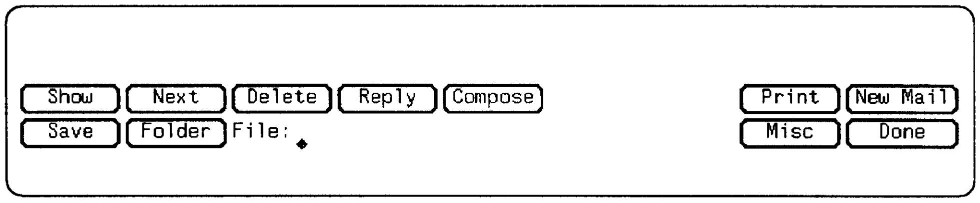
status msg. no. sender date time msg. length subject

- There are three status indicators for a letter. *N* means the letter is *new*. *U* indicates that the letter is *unread*; you've retrieved it from your mailbox—and possibly stored it away—without reading it. A blank status indicator means that you've retrieved and read the letter but you haven't done anything with it. > indicates the letter you're currently viewing.
- A *message number* assigned to each letter in the order received.
- The *sender* is the person who sent the letter.
- The *date* is the date the letter was sent.
- The *time* indicates the time it was sent.
- The *message length* is given in two parts: the first number is the number of lines in the letter, and the second number is the number of characters.
- The *subject* is assigned by the sender.

The Command Panel Window

The *command panel window* has buttons that control most Mail Tool functions. (These will be described in detail later.)

Figure 10-4 *Command Panel Window*



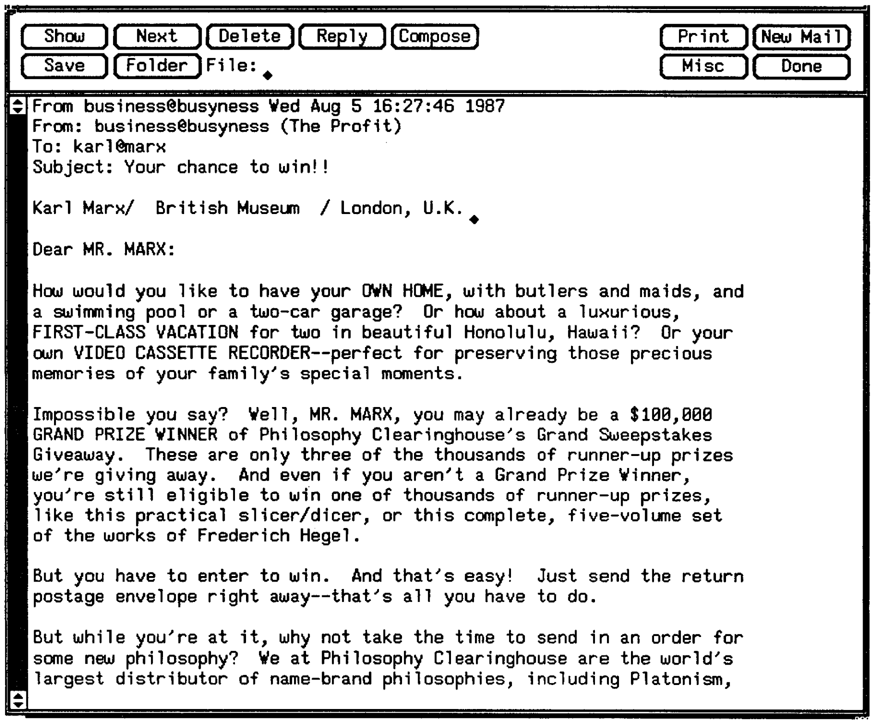
Pushing a Button

You push a button by using the mouse to position the pointer over a button in window — **Show** or **Compose**, for example — and then clicking with the left mouse button.

The Message Window

The window under the command panel window is called the *message window*; this is where Mail Tool displays the letters you receive.

Figure 10-5 *The Message Window*

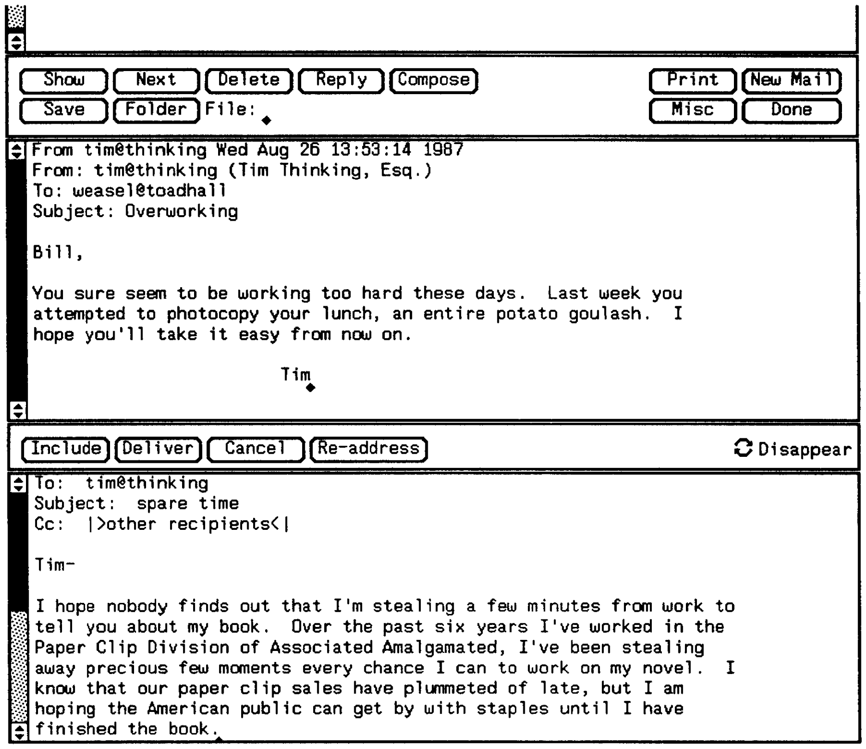


```
From business@busyness Wed Aug 5 16:27:46 1987
From: business@busyness (The Profit)
To: karl@marx
Subject: Your chance to win!!

Karl Marx/  British Museum  / London, U.K.

Dear MR. MARX:

How would you like to have your OWN HOME, with butlers and maids, and
a swimming pool or a two-car garage?  Or how about a luxurious,
FIRST-CLASS VACATION for two in beautiful Honolulu, Hawaii?  Or your
own VIDEO CASSETTE RECORDER--perfect for preserving those precious
memories of your family's special moments.

Impossible you say?  Well, MR. MARX, you may already be a $100,000
GRAND PRIZE WINNER of Philosophy Clearinghouse's Grand Sweepstakes
Giveaway.  These are only three of the thousands of runner-up prizes
we're giving away.  And even if you aren't a Grand Prize Winner,
you're still eligible to win one of thousands of runner-up prizes,
like this practical slicer/dicer, or this complete, five-volume set
of the works of Frederich Hegel.

But you have to enter to win.  And that's easy!  Just send the return
postage envelope right away--that's all you have to do.

But while you're at it, why not take the time to send in an order for
some new philosophy?  We at Philosophy Clearinghouse are the world's
largest distributor of name-brand philosophies, including Platonism,
```

The Composition Window

The composition window is where you write letters. Normally, you get a composition window by pushing either the **Compose** or the **Reply** button.

Figure 10-6 *The Composition Window*



```
From tim@thinking Wed Aug 26 13:53:14 1987
From: tim@thinking (Tim Thinking, Esq.)
To: weasel@toadhall
Subject: Overworking

Bill,

You sure seem to be working too hard these days.  Last week you
attempted to photocopy your lunch, an entire potato goulash.  I
hope you'll take it easy from now on.

                    Tim
```

```
To:  tim@thinking
Subject:  spare time
Cc:  |>other recipients<|

Tim-

I hope nobody finds out that I'm stealing a few minutes from work to
tell you about my book.  Over the past six years I've worked in the
Paper Clip Division of Associated Amalgamated, I've been stealing
away precious few moments every chance I can to work on my novel.  I
know that our paper clip sales have plummeted of late, but I am
hoping the American public can get by with staples until I have
finished the book.
```

10.6. Receiving Mail

There are three ways that you can be notified of incoming mail.

(1) Outside of Mail Tool, the SunOS operating system will give you the notification `You have new mail` in the window you're typing in.

(2) When Mail Tool is closed, the icon changes: the flag on the mailbox goes up, and a letter appears in the slot.

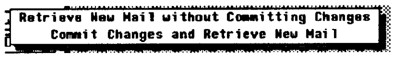
(3) When Mail Tool is open, the frame header at the top of Mail Tool will display `[New Mail]` when you receive a letter.

By using the Defaults Editor, you can make Mail Tool beep and flash the screen when it receives a letter. See Appendix B.

10.7. Retrieving New Mail and Committing Changes

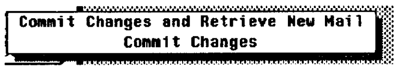
The **New Mail** button retrieves letters that you've received. Hold down the right mouse button to get the menu associated with the button and to choose from it.

NOTE *Clicking the left mouse button on a Mail Tool button always selects the topmost menu item for that button.*


Retrieve New Mail without Committing Changes
Commit Changes and Retrieve New Mail

The **New Mail** button has two different menus, depending on whether you are currently looking at your mailbox or at a mail file or folder. When you are looking at your mailbox, the **New Mail** button displays the menu at left.

If you push the **New Mail** button while looking at your mailbox, Mail Tool retrieves new letters *without committing changes*, which means you can still undelete letters you've deleted. On the other hand, if you choose 'Commit Changes and Retrieve New Mail,' all your deletions become permanent.



Note that if you are looking at a mail file or folder, then the **New Mail** menu changes. Pushing **New Mail** *always* commits changes when you are looking at mail. See Section 10.9 for more on folders.

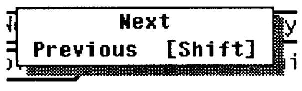
Choosing Letters

When you retrieve new mail, Mail Tool displays an updated header list and the current letter. Mail Tool puts that letter in the message window and puts a > in front of its header. (If you have unread letters when you push the **New Mail** button, Mail Tool will mark them with a U (unread).)

You can do an operation on a letter just by clicking on its header and pushing the appropriate button. You do not have to display the letter first. (The **Next** button is an exception.)

To read other new mail, you can move through the header list with two buttons, **Next** and **Show**.

Looking at the Next (or Previous) Letter



The **Next** button moves you to the next letter in the header list. (If you're looking at the last letter in the letter list, pushing the **Next** button displays the *previous* letter.)

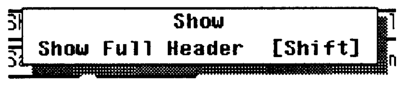
To look at the letter that comes before the current letter, either hold down the [Shift] key when you push the **Next** button or choose 'Previous' from the **Next** menu.

Looking at Other Letters With the Show Button

To display a letter, move the pointer to its header. Click the left mouse button to choose that header; then push the **Show** button. The letter will appear in the message window.

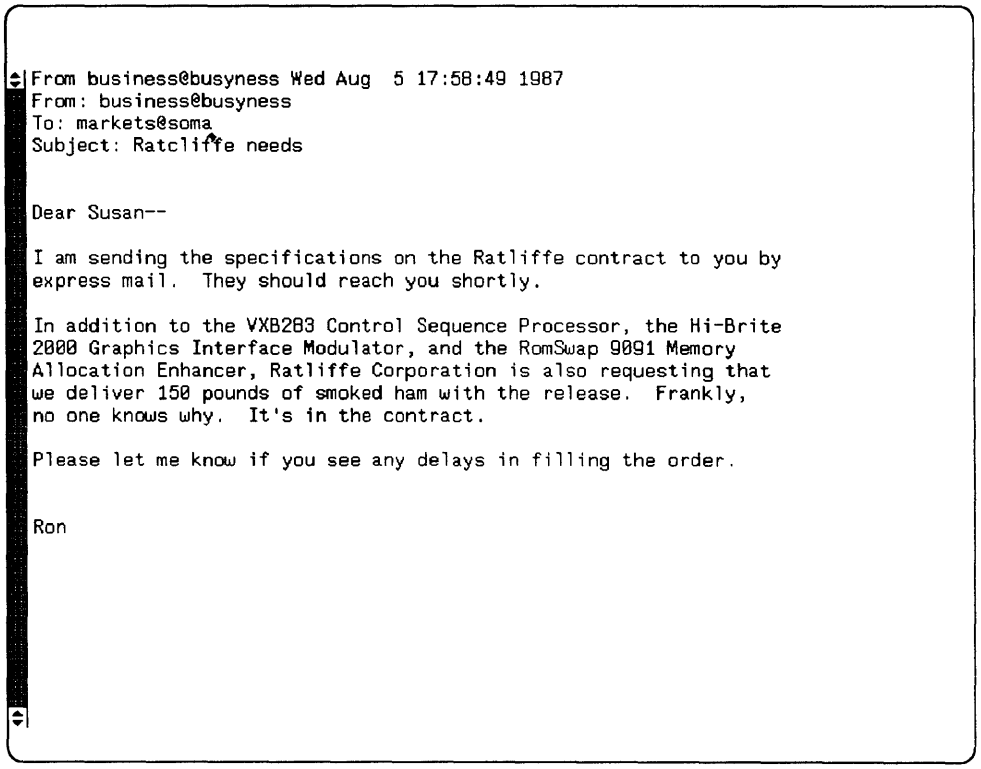
Displaying the Full Header

The header of a letter can contain a lot of unwanted information. Use the Defaults Editor to suppress the display of unwanted information. (This is explained in Appendix B.)



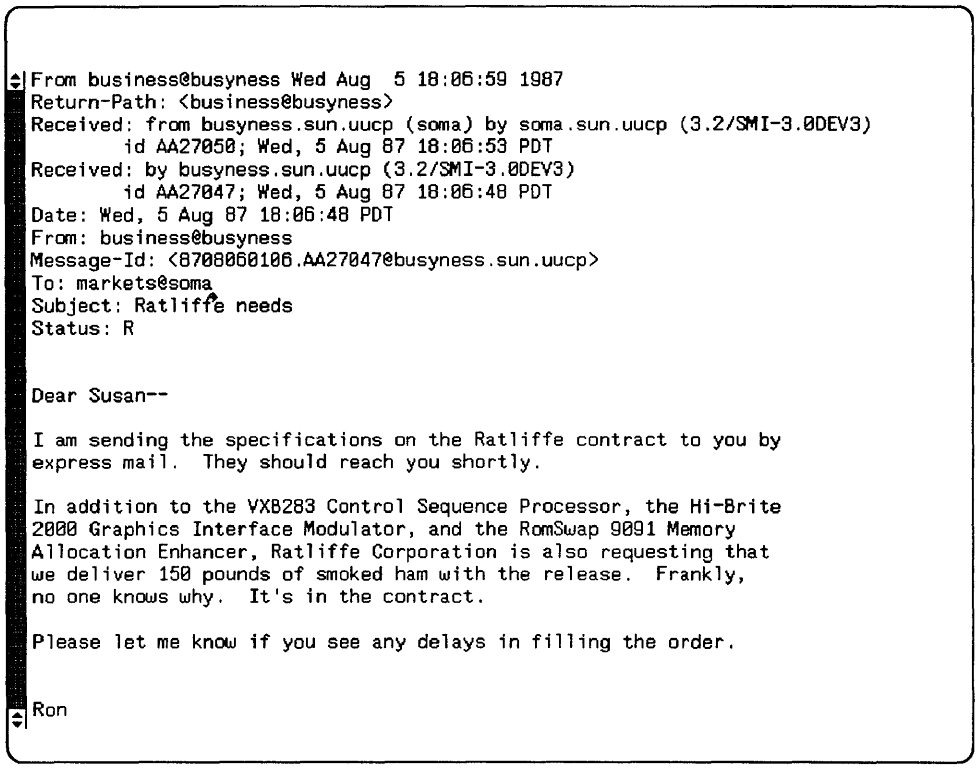
If you do suppress header information, you can still display it when you want to. The **Show** button has a 'Show Full Header' menu item (which you can get by using the [Shift] key on the **Show** button). Here is a letter, first displayed normally, and then with the 'Show Full Header' option:

Figure 10-7 *A Typical Mail Header*



```
From business@busyness Wed Aug  5 17:58:49 1987
From: business@busyness
To: markets@soma
Subject: Ratcliffe needs


Dear Susan--

I am sending the specifications on the Ratliffe contract to you by
express mail.  They should reach you shortly.

In addition to the VXB283 Control Sequence Processor, the Hi-Brite
2000 Graphics Interface Modulator, and the RomSwap 9091 Memory
Allocation Enhancer, Ratliffe Corporation is also requesting that
we deliver 150 pounds of smoked ham with the release.  Frankly,
no one knows why.  It's in the contract.

Please let me know if you see any delays in filling the order.


Ron
```

Figure 10-8 *Full Letter Header*



```
From business@busyness Wed Aug  5 18:06:59 1987
Return-Path: <business@busyness>
Received: from busyness.sun.uucp (soma) by soma.sun.uucp (3.2/SMI-3.0DEV3)
        id AA27050; Wed, 5 Aug 87 18:06:53 PDT
Received: by busyness.sun.uucp (3.2/SMI-3.0DEV3)
        id AA27047; Wed, 5 Aug 87 18:06:48 PDT
Date: Wed, 5 Aug 87 18:06:48 PDT
From: business@busyness
Message-Id: <8708060106.AA27047@busyness.sun.uucp>
To: markets@soma
Subject: Ratliffe needs
Status: R


Dear Susan--

I am sending the specifications on the Ratliffe contract to you by
express mail.  They should reach you shortly.

In addition to the VX8283 Control Sequence Processor, the Hi-Brite
2000 Graphics Interface Modulator, and the RomSwap 9091 Memory
Allocation Enhancer, Ratliffe Corporation is also requesting that
we deliver 150 pounds of smoked ham with the release.  Frankly,
no one knows why.  It's in the contract.

Please let me know if you see any delays in filling the order.


Ron
```

10.8. Printing Mail

Push the **Print** button to send a copy of a letter to a printer. (You can change the way you print mail out by using the Defaults Editor to change the `printmail` variable. See Appendix B.)

10.9. Saving Mail

In addition to composing and reading mail, you will also want to save and retrieve it.

Here's *one* way of saving a letter (other ways — employing folders — are discussed below):

- In the command panel window note the word `File:`
- After the colon, type the name of the file in which you want the letter held.
- Push the left mouse button on the **Save** button. *Or* hold down the right mouse button over the **Save** button to view the menu options and to select 'Copy.'

Saving deletes the letter from the mailbox and puts it into the file you specified. *Copying* leaves the letter visible in the mailbox, while putting a copy into the specified file.

Suppose you're saving a letter about Raymond Chandler's novel *The Big Sleep* into the file `classic_mysteries`. If you just type

`classic_mysteries`

after `File:` Mail Tool will assume that `classic_mysteries` is located in its current directory. If you want `classic_mysteries` to be in a different directory, say `/home/medici/mail`, you can type in the file's whole name, `/home/medici/mail/classic_mysteries`, after `File:`

Setting Up Folders and the Folder Directory

But you can dispense with typing full pathnames if you save to Mail Tool *folders*: folders are files stored in a *folder directory.*

NOTE *All folders are files, but not all files are folders. A folder directory is a directory that contains folders. . . .*

Setting up a folder directory in which to store your letters is a three-step process:

- First, type `mkdir` *directoryname* on a command line (name the directory anything you like).
- Then use the Defaults Editor to alter the the Mail Tool settings. At the `Set/folder` variable type after the colon the pathname of the directory you created in the first step. Don't forget to *save* your change before quitting the Defaults Editor.
- Choose 'Source .mailrc' from the **Misc** button menu. See Section 10.19.

For example, if you've assigned `/home/medici/mail` as your folder directory, that's where your folders will be stored.

Naming Folders

You name a folder by typing a + and a name after `File`: The + attaches the folder to the folder directory.

Thus, still using the example above, the foldername `+classic_mysteries` is interpreted by Mail Tool as `/home/medici/mail/classic_mysteries`.

The + is invisible outside `mail` and Mail Tool.

Selecting and Retrieving Files and Folders

To select a folder and to retrieve a folder or file, you use the **Folder** button:

- To select a *folder*:

  Hold down the right mouse button over the **Folder** button — this will give you a menu of all the folders in your folder directory. Choose a folder: its name will appear after `File:` (You can also select a folder by typing its name after `File:`)

- To select a *file*:

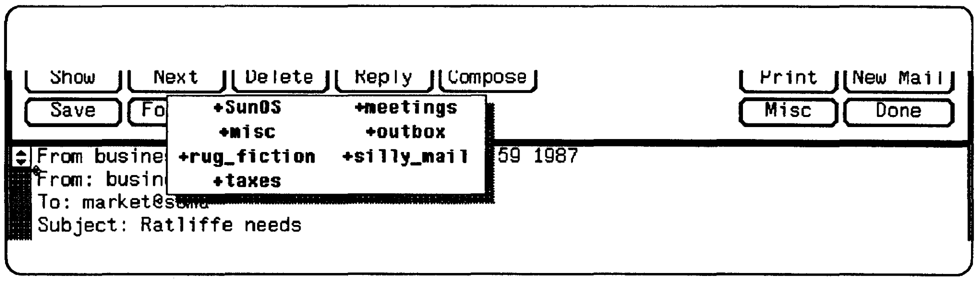
  Type the full pathname after `File:`

- To *retrieve* a folder or file:

  Once the folder or file has been selected, use the left mouse button to push the **Folder** button. The letters in that folder will appear in Mail Tool with the usual header information.

NOTE *To exit a mail file or folder, use the left mouse button to push the **New Mail** or **Done** button. Or select another folder or mail file.*

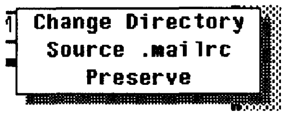
Figure 10-9 *A Folder Directory*



```
From busine                          59 1987
From: busin
To: market@s
Subject: Ratliffe needs
```

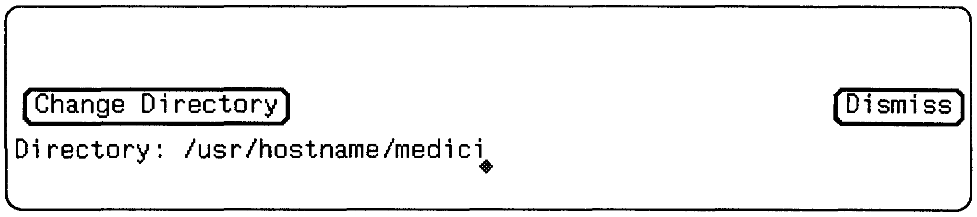
A final word about folders: you can get a menu of just the folders you've used during this Mail Tool session by holding down the right mouse button over `File:` If you work with a few folders over and over, this can save you time.

Changing Directories With the Misc Button



The 'Change Directory' menu option of the **Misc** button lets you change the directory that Mail Tool is active in. When you choose 'Change Directory,' a pop-up window appears giving you the current directory and asking you for the name of the new directory. Type in the name of the directory where you want to save and retrieve letters.

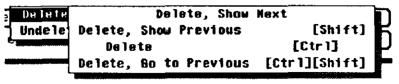
Figure 10-10 *The Change Directory Pop-up Window*



```
Directory: /usr/hostname/medici
```

NOTE *'Change Directory' does not affect where Mail Tool looks for folders: the folder directory remains unchanged.*

10.10. Deleting (and Undeleting) Mail


Delete
Undelete
Delete, Show Next
Delete, Show Previous [Shift]
Delete [Ctrl]
Delete, Go to Previous [Ctrl][Shift]

When using Mail Tool you'll want to delete letters from time to time.

To delete a letter from the mailbox or from a folder, find it in the header list and click the left mouse button to select it, and then push the **Delete** button.

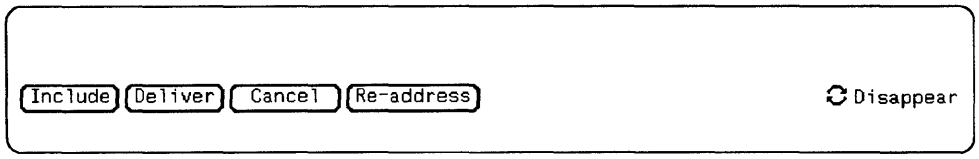
A deleted letter can be recovered by using the 'Undelete' option of the **Delete** menu. A letter can be undeleted at any time — until you commit changes (usually when retrieving new mail or or finishing a Mail Tool session). If you commit changes, you remove deleted letters from your mailbox and make your changes permanent.

10.11. Composing Mail

To compose a letter, push the **Compose** button with the left mouse button. A composition window will appear. You can type and manipulate text in this window in the same ways as in other SunView text windows.

You can open additional composition windows. These will appear as independent windows that you can open, close, and resize without affecting the rest of Mail Tool. They disappear when you quit Mail Tool. You can make even the initial composition window a pop-up window by using the Defaults Editor to set the `alwaysusepopup` variable to 'Yes.' See Appendix B for more on this.

Figure 10-11 *Composition Window Buttons*



Inserting Names and Addresses

Every composition window has pre-set fields for recipients ('To:'), subject ('Subject:'), and other recipients ('Cc:') of your letter.

Type in the appropriate information in each field. Type [Ctrl-Tab] to move from one field to the next.

NOTE *An empty field is not sent as part of your letter.*

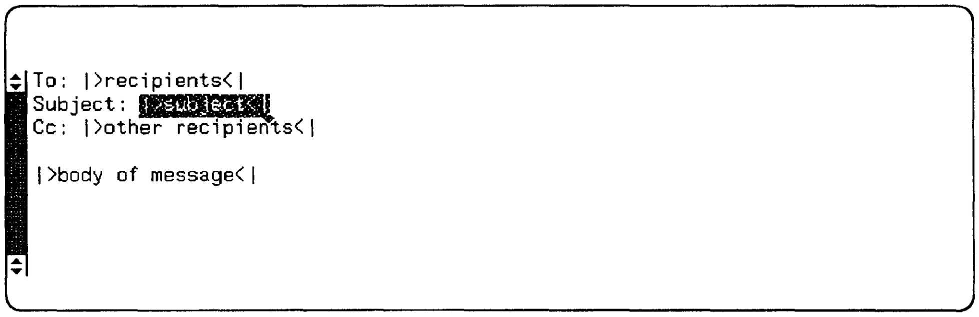
You can send the letter to as many recipients as you wish: just leave a space or a comma between names after 'To:' and 'Cc:'. (Note that the **Compose** button menu gives you the option of not having a 'Cc:' field.)

And you can use the Defaults Editor to have a 'Bcc:' (blind carbon) field pre-set: change the `Set/askbcc` option to 'Yes.' Recipients of your letters will not see the addressees of the blind carbon copies, but the addressees of carbon copies will be visible to all.

Sending a Letter Directly to a File or Folder

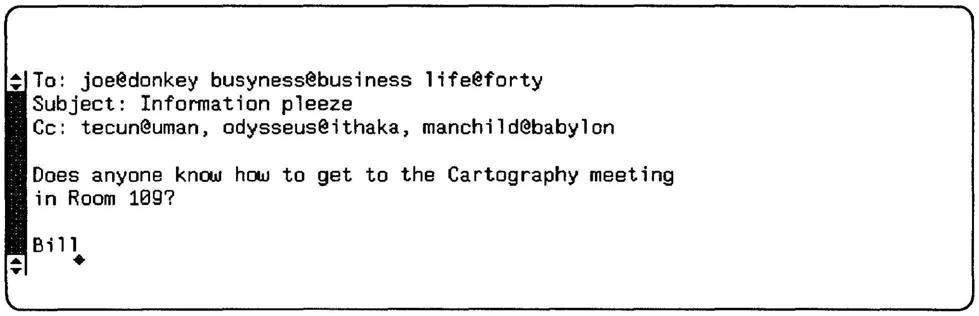
You can also send a copy of a letter directly to a file or mail folder. Simply type the file pathname or folder name in either the `Cc:` or the `Bcc:` field.

Figure 10-12 *Subject and Address Fields in a Composition Window*



```
To: |>recipients<|
Subject: |>subject<|
Cc: |>other recipients<|

|>body of message<|
```

Figure 10-13 *Multiple Addresses*

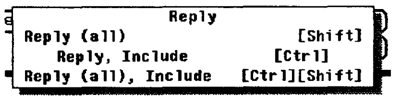

```
To: joe@donkey busyness@business life@forty
Subject: Information pleeze
Cc: tecun@uman, odysseus@ithaka, manchild@babylon

Does anyone know how to get to the Cartography meeting
in Room 109?

Bill
```

Using Word-Wrap

While you *can* set an automatic word-wrap for Mail Tool composition windows (as in other text windows), it's probably not a good idea. If the person receiving your letter isn't using word-wrap, the lines of the letter could run out the window.

On the other hand, if you receive a ragged letter from someone, you can clean it up by selecting the text while holding down the [Ctrl] key and then choosing 'Extras⇒Format' from the text menu in Mail Tool's message window.

10.12. Replying to Mail



To reply to a letter you've received, select the letter in the header list and then use the left mouse button to push the **Reply** button. A composition window will appear — preaddressed to the sender of the letter you're answering.

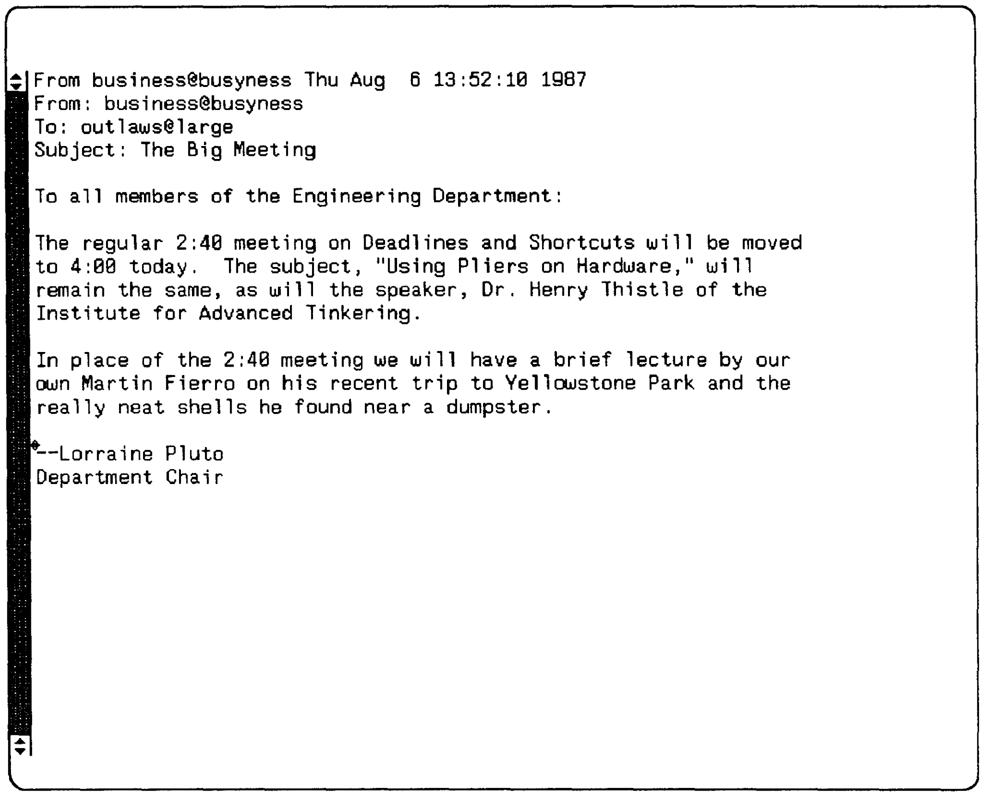
By selecting 'Reply (all)' or 'Reply (all), Include' **Reply** button items, you can send your reply to everyone else who received the letter as well as to its author.

10.13. Including a Letter or File

Mail Tool lets you forward letters by putting a copy of a letter you received into a letter you send.

Suppose you received this letter, and you wanted to send a copy of it to `joe@donkey`:

Figure 10-14 *A Received Letter*

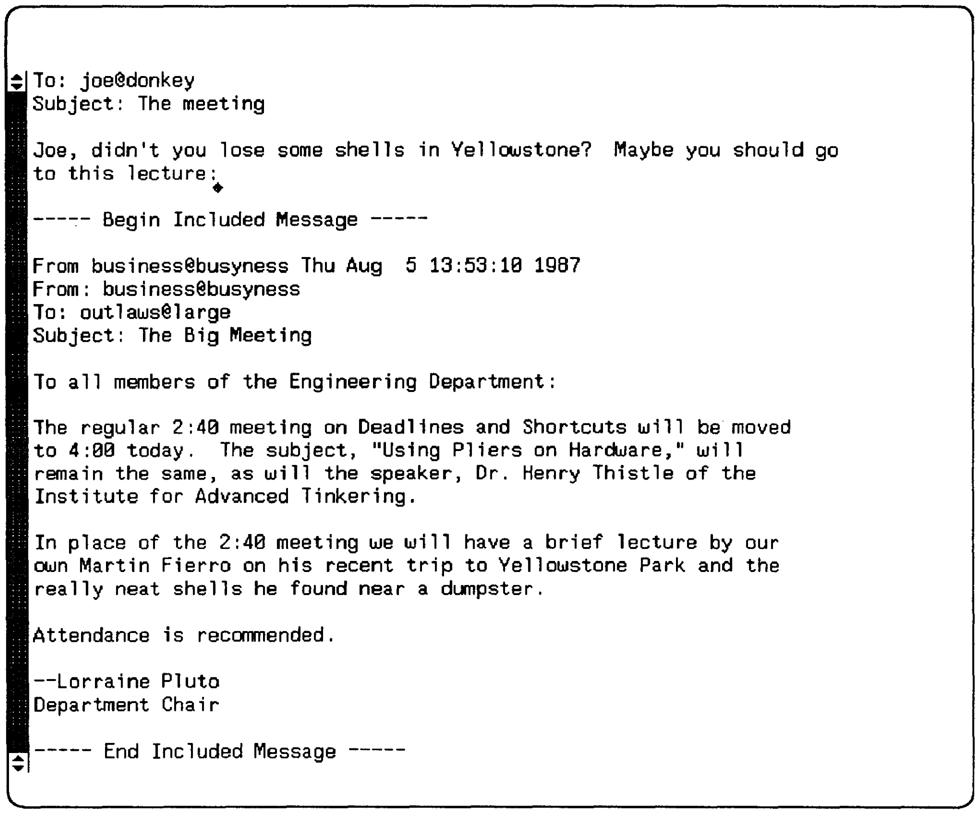


```
From business@busyness Thu Aug  6 13:52:10 1987
From: business@busyness
To: outlaws@large
Subject: The Big Meeting

To all members of the Engineering Department:

The regular 2:40 meeting on Deadlines and Shortcuts will be moved
to 4:00 today.  The subject, "Using Pliers on Hardware," will
remain the same, as will the speaker, Dr. Henry Thistle of the
Institute for Advanced Tinkering.

In place of the 2:40 meeting we will have a brief lecture by our
own Martin Fierro on his recent trip to Yellowstone Park and the
really neat shells he found near a dumpster.

--Lorraine Pluto
Department Chair
```

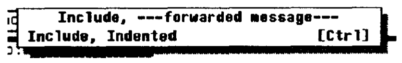
Using the Include Button

First, bring up the composition window with **Compose** or **Reply**. Then make sure that the cursor in the composition window is where you want the inserted letter to go. Push the **Include** button.

Figure 10-15 *Including a Letter*

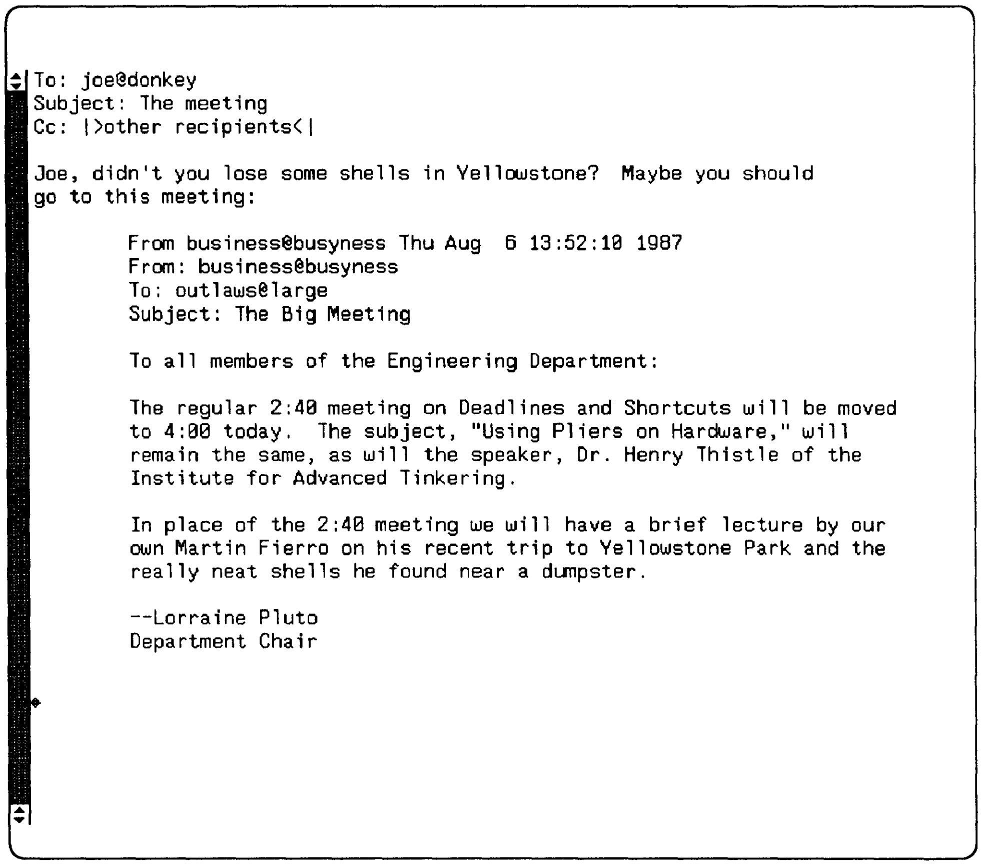


```
To: joe@donkey
Subject: The meeting

Joe, didn't you lose some shells in Yellowstone?  Maybe you should go
to this lecture:

----- Begin Included Message -----

From business@busyness Thu Aug  5 13:53:10 1987
From: business@busyness
To: outlaws@large
Subject: The Big Meeting

To all members of the Engineering Department:

The regular 2:40 meeting on Deadlines and Shortcuts will be moved
to 4:00 today.  The subject, "Using Pliers on Hardware," will
remain the same, as will the speaker, Dr. Henry Thistle of the
Institute for Advanced Tinkering.

In place of the 2:40 meeting we will have a brief lecture by our
own Martin Fierro on his recent trip to Yellowstone Park and the
really neat shells he found near a dumpster.

Attendance is recommended.

--Lorraine Pluto
Department Chair

----- End Included Message -----
```



Choosing the 'Include, Indented' option of the **Include** menu indents the included letter, as in the example below. The included letter is inserted at the cursor.

Figure 10-16 *Including a Letter, Indented*



```
To: joe@donkey
Subject: The meeting
Cc: |>other recipients<|

Joe, didn't you lose some shells in Yellowstone?  Maybe you should
go to this meeting:

        From business@busyness Thu Aug  6 13:52:10 1987
        From: business@busyness
        To: outlaws@large
        Subject: The Big Meeting

        To all members of the Engineering Department:

        The regular 2:40 meeting on Deadlines and Shortcuts will be moved
        to 4:00 today.  The subject, "Using Pliers on Hardware," will
        remain the same, as will the speaker, Dr. Henry Thistle of the
        Institute for Advanced Tinkering.

        In place of the 2:40 meeting we will have a brief lecture by our
        own Martin Fierro on his recent trip to Yellowstone Park and the
        really neat shells he found near a dumpster.

        --Lorraine Pluto
        Department Chair
```

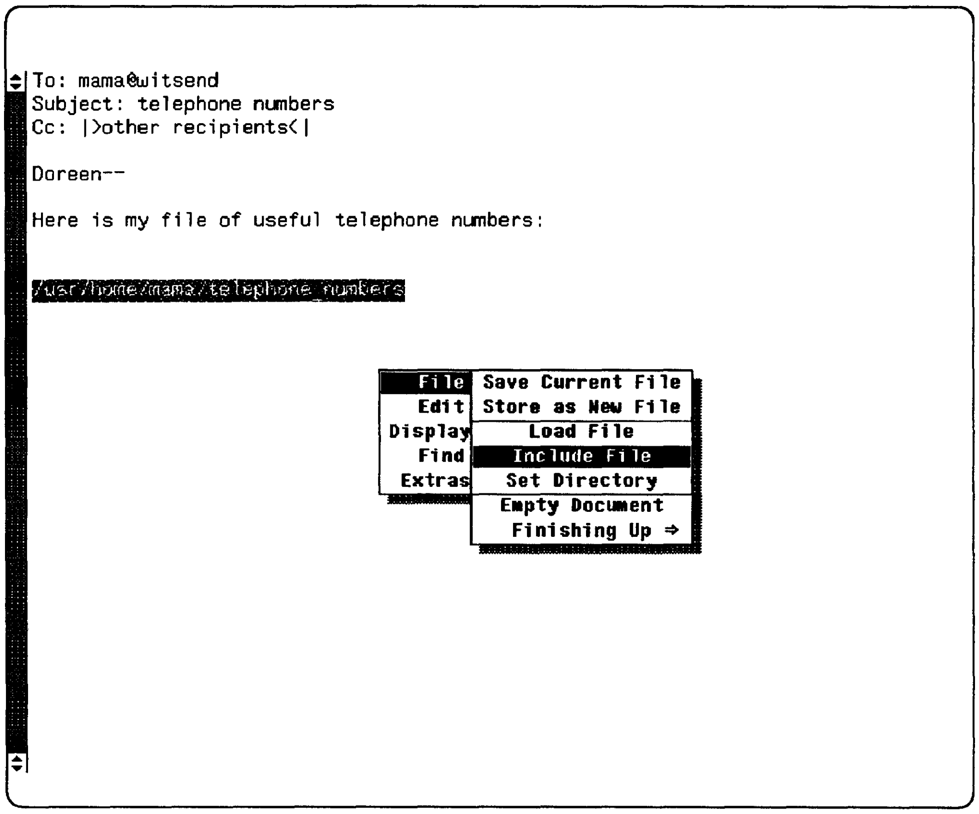
Using the Compose and Reply Buttons

The **Compose** and **Reply** buttons have 'Include' as a menu option. This option lets you include a letter directly when you bring up a composition window, instead of using the **Include** button.

Using 'Include File'

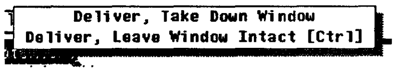
You can also include any other file in a letter by using the Text Edit menu item 'Include File.' Type the filename, select it while holding down the Ctrl key, and then choose 'Include File' from the menu. (The filename itself is *not* included.)

Figure 10-17 *The Text Edit Menu in Mail Tool*



```
To: mama@witsend
Subject: telephone numbers
Cc: |>other recipients<|

Doreen--

Here is my file of useful telephone numbers:


/usr/home/mama/telephone_numbers
```

10.14. Delivering a Letter



Deliver, Take Down Window
Deliver, Leave Window Intact [Ctrl]

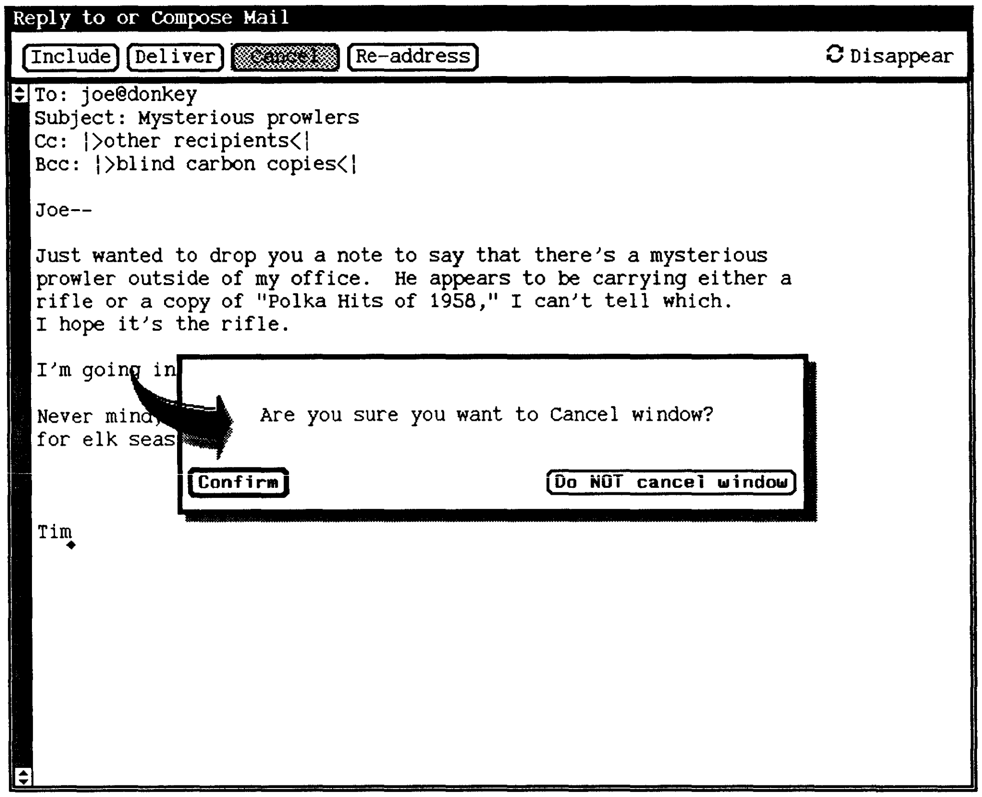
Once you've written your letter, you send it with the **Deliver** button. Be sure that your letter's ready to send before you push **Deliver** — once a letter is sent, you can't take it back!

The **Deliver** menu allows you to send a letter but leave the composition window intact; that is, with the letter still in it. This is useful if you want to send the letter off to someone else (possibly editing it first) or to keep a copy for yourself.

10.15. Canceling a Letter

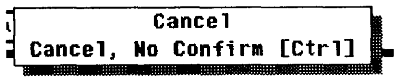
You can cancel a letter only *before* you send it. When you press the **Cancel** button, Mail Tool will ask you to confirm the cancellation.

Figure 10-18 *Confirming a Cancellation*



```
To: joe@donkey
Subject: Mysterious prowlers
Cc: |>other recipients<|
Bcc: |>blind carbon copies<|

Joe--

Just wanted to drop you a note to say that there's a mysterious
prowler outside of my office.  He appears to be carrying either a
rifle or a copy of "Polka Hits of 1958," I can't tell which.
I hope it's the rifle.

I'm going in

Never mind
for elk seas

Tim
```

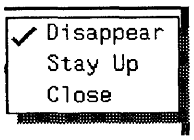
Are you sure you want to Cancel window?



You can avoid this confirmation query by choosing 'Cancel, No Confirm' from the **Cancel** menu or by holding down the [Ctrl] key when you push the **Cancel** button.

You can modify Mail Tool so that it will not ask you to confirm certain irreversible operations such as quitting windows or canceling a composition. You do this by using the Defaults Editor to turn on the `expert` variable. See Appendix B.

10.16. Reusing a Composition Window

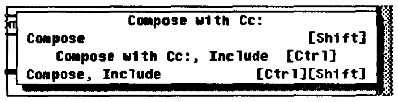


Normally, when you deliver or cancel a letter, the composition window folds up and goes away. If you know you're going to want to write another letter, though, you can set the composition window to stay up after you deliver or cancel a letter. You use the **Disappear** button to do this.

When the composition window is a pop-up window, you can set it to close itself to iconic form after canceling or delivering. (But the window will stay up if you choose 'Leave Window Intact' from the **Deliver** menu, regardless of how the **Disappear** cycle is set.)

When the **Disappear** button is set to 'Stay Up' and you cancel or deliver a letter, the composition window goes blank. To put the address template back into the window, push the **Re-address** button.

Note that the **Cancel** button changes to **Clear** when the **Disappear** cycle is set to 'Stay Up.'


Compose with Cc:
Compose [Shift]
Compose with Cc:, Include [Ctrl]
Compose, Include [Ctrl][Shift]

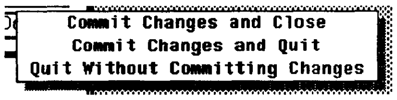
10.17. Closing Mail Tool

When Mail Tool is closed, it reverts to its iconic form, though it continues to run.

Closing using SunView:

> You can close Mail Tool the way you close any other SunView window. The fastest way to close Mail Tool is to press the [Open] key on the keyboard.
>
> Closing this way does not commit your changes. That means that any letters you have deleted will not be permanently removed, and when you open Mail Tool again you'll still be able to undelete them. However, they'll still be in your mailbox, taking up room — if you always open and close Mail Tool this way, you may soon have a mailbox full of temporarily deleted letters filling up your file system. It's a good idea to commit your changes now and then, just to keep your mailbox a manageable size.
>
> Closing Mail Tool from the SunView menu also means that Mail Tool will not automatically look for new mail when you open it again. This saves time, but you will have to push the **New Mail** button when you want to search for new mail.

Closing using **Done**:



> The other way to close Mail Tool is by using the **Done** button. When you push **Done**, Mail Tool commits all changes before closing itself. The next time you open Mail Tool, it will automatically look for new mail.

10.18. Quitting Mail Tool

Quitting Mail Tool renders it inactive — the window closes and the icon disappears.

Quitting using SunView:

> You can quit Mail Tool as you would quit any SunView window (see the *SunView User's Guide*). For example, you can hold down the right mouse button over the top border of the Mail Tool window and then choose 'Quit' from the menu. If you do so, Mail Tool will commit changes before it disappears (changes will be reflected the next time you start Mail Tool).

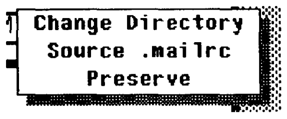
Quitting using **Done**:

You can also use the **Done** button to quit. There are two ways to do this:

- Choosing 'Commit Changes and Quit' from the **Done** menu commits all your changes before quitting Mail Tool.

- Choosing 'Quit Without Committing Changes' also quits Mail Tool. When you restart Mail Tool, you'll find that any letters you've deleted without committing will still be there, as though you'd never deleted them.

10.19. Changing Mail Tool: Mysteries of the Misc Button



The 'Source .mailrc' option modifies Mail Tool's behavior. Recall that the `.mailrc` file contains settings affecting Mail Tool. Normally, when you modify `.mailrc` you have to quit and restart Mail Tool to see your changes take effect. Choosing 'Source .mailrc' allows you to incorporate changes to `.mailrc` while Mail Tool is running.

There is a limitation on 'Source .mailrc,' however: If you change the setting of a variable from ''not set'' to ''set'' (from off to on), then 'Source .mailrc' puts the change into effect. But if you change the variable from ''set'' to ''not set'' (from on to off), then it won't. So, when changing variable from ''set'' to ''not set,'' quit and restart Mail Tool after changing `.mailrc`. For a complete discussion of Mail Tool variables and how to change them, see Appendix B.

11

Mail

11.1. What Is `mail`?

`mail` is a SunOS program for sending and receiving electronic mail. It provides facilities for reading, writing, sending, receiving, saving, and deleting letters. `mail` is not window-based; thus it can be run on any terminal.

Electronic mail (or *e-mail*) is modeled on the traditional post: letters addressed to others on the system or network are received by users in electronic mailboxes. As may be expected, e-mail is considerably quicker and less subject to the vagaries of weather; still, some correspondents will miss the colorful stamps.

"`mail` Basics" gives you the absolute minimum necessary to start `mail`, send a letter, read letters, and quit the program. The remainder of the chapter explores many useful features of `mail` and includes discussion on using the `vi` text editor for writing and editing letters.

11.2. `mail` Basics

In this section you'll learn just enough about `mail` to get by. In later sections you'll learn about features and functions that will greatly enhance your ability to use e-mail.

Starting `mail`

Start `mail` by typing the following command at a prompt and then pressing [Return]:

```
venus% mail
```

If you don't have any mail waiting for you, your terminal will display the message:

```
No mail for user
venus%
```

Sending Yourself a Letter

To see at a glance how `mail` works, you can begin by sending yourself a letter. At the prompt, give the `mail` command again, but this time include your address (your log-in plus machinename). For example, if your log-in were "aphrodite" and your machinename were "venus," your address would be "aphrodite@venus." (The @ symbol is read as "at.") You may be able to use just your log-in on a local network— consult your system administrator when in doubt.

```
venus% mail aphrodite@venus
```

The program will reply with a `Subject:` line:

```
venus% mail aphrodite@venus
Subject:
```

If you like, type in a word or two here about the content of the letter you're sending yourself and press [Return]. Now type the body of the letter; use short lines and press [Return] at the end of each line. (Note that you can only make corrections as you go by back-spacing and retyping lines *before* you press [Return].)

Your sample letter might look something like this (the spaces between lines are made by pressing [Return] twice):

```
venus% mail aphrodite@venus
Subject: banishment of solitude

Dear Aphrodite,

Though I'm not enamored of the thought of
writing to myself--I'd love to hear from Eros--
it is a way of passing the time.

Maybe I'll read a thriller tonight.

See you soon,

Aphrodite
```

Send your letter by pressing [Return] to start a new line and then pressing [Ctrl-D]. After your letter has been sent, the system will return you to a command prompt.

Reading a Letter

To read your letter, give the `mail` command again. Your screen will probably look something like this:

```
venus% mail
Mail version SMI 4.0 Mon Apr 24 17:36:41 PDT 1989 Type ? for help
"/usr/spool/mail/aphrodite": 1 message

>N  1 aphrodite@venus   Fri Jul 14 12:01 21/453 banishment of

&
```

Following the information that identifies the version of `mail` you're running and the mailbox where your incoming mail is deposited, there is the header of the letter you sent yourself.

Every letter is assigned a number as it's received: Aphrodite's letter to herself is shown as letter number 1. (Header information will be analyzed in Section 11.3.)

To read a letter, type its number at the `mail` *prompt:*

```
venus% mail
Mail version SMI 4.0 Mon Apr 24 17:36:41 PDT 1989 Type ? for help
"/usr/spool/mail/aphrodite": 1 message

>N  1 aphrodite@venus   Fri Jul 14 12:01 21/453 banishment of

& 1

To: aphrodite@venus
From: aphrodite@venus
Subject: banishment of solitude

Dear Aphrodite,

Though I'm not enamored of the thought of
writing to myself--I'd love to hear from Eros--
it is a way of passing the time.

Maybe I'll read a thriller tonight.

See you soon,

Aphrodite

&
```

Sending Someone Else a Letter

Now that you know how to send and read a letter, you can send to other users by giving the `mail` command followed by a username.

```
venus% mail user@machine
```

If you've already started `mail`, you can use the same command at the `mail` prompt:

```
& mail user@machine
```

Quitting `mail`

When you're finished using `mail`, you can quit the program by typing `q` at the `mail` prompt and then pressing [Return]:

```
& q
```

You will then see the message,

```
Saved one message in home_directory/mbox.
```

Whenever you quit `mail` after reading a message, it saves the letter in the `mbox` file in your home directory, unless you delete it first.

11.3. Reading and Deleting Letters

If you have mail, `mail` notifies you each time you log in with the message

```
You have mail
```

or

```
You have new mail
```

To read your letters, type `mail` at a command prompt and press [Return]. If there's no mail waiting for you, you'll see the message,

```
No mail for aphrodite
```

Otherwise, you'll see something like this:

```
venus% mail
Mail version SMI 4.0 Mon Apr 24 17:36:41 PDT 1989 Type ? for help
"/usr/spool/mail/aphrodite": 3 messages 1 new 1 unread

    1 aphrodite@venus    Fri Jul 14 12:01 21/453 banishment of
 U  2 maigret@paris      Fri Jul 14 18:31 19/353 pipe perdue
>N  3 marlowe@baycity    Sun Jul 16 23:59 14/280 getting sleep

&
```

The `mail` program displays information about itself (version number and date) and instructions for getting help (`Type ? for help`). For more on help, see Section 11.8.

On the next line, `mail` specifies the location of your mailbox (`/usr/spool/mail/aphrodite`), how many letters you've received (`3 messages`), and their status (`1 new 1 unread`).

Next, `mail` shows a numbered list of the letters in your mailbox. Reading across, the columns in each line specify:

- *Status:* indicates whether a letter is new (`N`), unread (`U`), or read (no symbol). > at the beginning of a line indicates the *current letter*. Deleted letters are marked with an `*`.
- *Number:* indicates order in which letter was received.

- *Sender:* indicates name of user (and usually machine) letter came from.
- *Time:* indicates date and time letter was sent.
- *Size:* indicates number of lines/number of characters in letter.
- *Subject:* indicates sender-designated subject of letter.

Selecting a Letter

To see the current letter (marked with >), press [Return]. Pressing [Return] again will display the next letter. To read *any* letter in the list, type its number and press [Return]. In any case, the letter will look something like this:

```
Message 3:
From marlowe@baycity Sun Jul 16 23:59 1989
Return-Path: <marlowe@baycity>
Received: by venus.sun.com (4.0/SMI-4.0)
    id AA12623; Sun, 16 Jul 89 23:59 PDT
Date: Sun, 16 Jul 89 23:59 PDT
From: marlowe@baycity (Philip Marlowe)
Message-Id: <8910232253.AA12623@venus.sun.com>
To: aphrodite@venus
Subject: getting sleep
Status: R

Like I always say, when in doubt, have a man
come through the door with a gun in his hand.

Sweet dreams,

Marlowe

&
```

The header of the letter contains more information than you'll probably find desirable. Appendix B shows you how to set up `mail` to suppress unwanted header information.

The `Date:`, `From:`, `To:`, and `Subject:` lines have been seen before. The `Return-Path:` line indicates the address used to return undeliverable mail; `Received:` indicates the machine, letter identification, and arrival time for each machine in the letter's network path; and `Message-Id:` provides identification information.

Deleting (and Undeleting) Letters From the Mailbox

To delete a letter, use the command form

```
d number
```

at a `mail` prompt.

For example, to delete the second letter, you would give this command (from inside `mail`):

```
& d 2
```

You can also delete several letters a time. To delete letters 1 *and* 3, you would give the command:

```
& d 1 3
```

To delete letters 1 *through* 3, give the command:

```
& d 1-3
```

Before quitting `mail` you can *undelete* letters you've removed from your mailbox. Use the command form

`u` *`number`*

followed by [Return]. To undelete the second letter, you would give this command:

```
& u 2
```

Note that all deletions are made permanent when you quit `mail` with the `q` command; that is, deleted letters can no longer be undeleted.

You can, however, quit `mail` with another command that leaves your mailbox intact— quitting with `x` will leave read letters left marked with a `U`, deleted letters undeleted, etc.

```
& x
```

11.4. Printing a Hard Copy

You can print out a hard copy of a letter by by piping a letter to a printer command. To do this, use the command form

`|`*`number`* `lpr`

at a `mail` prompt. (The | symbol is called "pipe." See Sections 4.14 for more on printing and 7.6 for more on pipes.)

For example, to print a copy of letter 2, you would type:

```
& |2 lpr
```

Press [Return].

If a letter number is not specified, `mail` will print the current letter.

11.5. Sending Letters

You can send mail from either the system prompt or the `mail` prompt. In either case, send a letter by giving the `mail` command followed by the address of the user; press [Return].

You can send the same letter to multiple addressees by typing in all their addresses —separated by a space or a comma— after the `mail` command. For example, if you wanted to send the same letter to ''hammett@glasskey'' and ''nisbet@frisco,'' you would give this command at a system prompt:

```
venus% mail hammett@glasskey nisbet@frisco
```

Or you could give the same command at a `mail` prompt:

```
& mail hammett@glasskey nisbet@frisco
```

Type your letter, pressing [Return] at the end of each line.

When you've finished, send the letter by typing [Ctrl-D] on a new line. (You can also send a letter by typing a period on a new line.)

Canceling an Unsent Letter

You can cancel a letter *before* it's sent by pressing [Ctrl-C] twice.

Adding Carbon and Blind Carbon Copies

Before sending a letter, you can specify that ''carbon copies'' be sent to other than the main addressees. You can also have ''blind carbons'' sent. (The difference is that recipients of your letter can read the addresses of the carbons but not of the blind carbons.)

Many people send themselves carbons or blind carbons in order to retain a copy for their own record. See below for ways to save mail in files and folders (special files for mail).

There are three ways to send a carbon copy. First, after you've finished typing the body of your letter, use the command form

```
~c address(es)
```

Type this command on a new line and separate addresses with a space. Press [Ctrl-D] to send the letter:

```
~c paretsky@shopping parker@boston
```

Or you can modify `mail` to make it prompt you with a `Cc:` *after* you've pressed [Ctrl-D]. See Appendix B on how to modify `mail`.

A `Cc:` line is also created by the `~h` command, which displays the entire header of the letter. `~h` prompts you with `To:`, `Subject:`, `Cc:`, and `Bcc:` lines, one line at a time. Blank lines can be filled in; filled lines can be retyped. As with other tilde commands, always give the `~h` command on a new line.

Inserting a Copy of a Letter or File

You can insert a copy of any letter in your mailbox into the letter you're writing. Likewise, you can insert a copy of any text file.

Inserting a Letter

The command form to insert a letter is

```
~m number
```

where *number* is the number of the letter to be inserted.

For example, to send a letter to ''maigret@paris'' that included a copy of the letter received from ''marlowe@baycity,'' you would:

- On a new line give the command `~m 3` and then press [Return].

  `mail` will display the message,

  ```
  Interpolating: 3 (continue)
  ```

  You won't see the text of message 3, but the recipient will. You can go on with your letter after `continue` or you can send it as is.

- To see the complete letter, interpolation included, give the commmand `~p`.

On your screen, the process would look like this:

```
venus% mail
Mail version SMI 4.0 Mon Apr 24 17:36:41 PDT 1989 Type ? for help
"/usr/spool/mail/aphrodite": 3 messages 1 new 1 unread

>   1 aphrodite@venus   Fri Jul 14 12:01 21/453 banishment of
    2 maigret@paris     Fri Jul 14 18:31 19/353 pipe perdue
    3 marlowe@baycity   Sun Jul 16 23:59 14/280 getting sleep

& mail maigret@paris
Subject: unsolicited advice

Mon cher Maigret, voici les conseils de Marlowe
sur des problemes structuraux du roman policier.

Tres amicalement,

Aphrodite
~m 3
Interpolating: 3
(continue)
~p

------
Message contains:
To: maigret@paris
Subject: unsolicited advice

Mon cher Maigret, voici les conseils de Marlowe
sur des problemes structuraux du roman policier.

Tres amicalement,

Aphrodite

>From marlowe@baycity Sun Jul 16 23:59 1989
>From: marlowe@baycity (Philip Marlowe)
>To: aphrodite@venus
>Subject: getting sleep
>
>Like I always say, when in doubt, have a man
>come through the door with a gun in his hand.
>
>Sweet dreams,
>
>Marlowe
>
(continue)
```

Inserting a File

You can also insert a copy of any text file into a letter. Use the command form

~r *filename*

as you're writing a letter.

Replying to a Letter

Reply to mail by giving the command

```
r number
```

at a `mail` prompt. (If you leave off the letter number, `mail` assumes that you're replying to the current letter.)

For example, to reply to the sender of letter 2, you would give the command:

```
& r 2
```

CAUTION **The commands `r` and `~r` produce very different results!**

`mail` will automatically address your letter as well as supply a `Re: Subject:` line that echoes the original `Subject:` line. Send your reply like any other letter.

`R` is a variant of the reply command that sends your reply to *all* recipients of the original letter as well as to its sender.

Note that you can insert a letter into your reply as shown in the previous section. To insert a copy of the letter you're replying to, just give the command `~m` without a letter number.

11.6. Saving and Retrieving Letters

In addition to sending and receiving letters, you'll also want to save and retrieve them for later use. In `mail` you can save letters by appending them to regular text files; you can also append letters to special files called folders. Both methods are discussed below.

`mail` makes a distinction between *saving* letters and *copying* them: saving removes a letter from the mailbox and appends it to a file or folder; copying leaves a letter in the mailbox and appends a copy to a file or folder.

Saving and Copying Letters in Files

To save a letter into a file, the command form is

```
s number filename
```

where *number* is the number of the letter to be saved and *filename* is the file where it's to be saved.

For example, to save letter 3 into a file called `~/fictional_professions/detectives`, you would give the command:

```
& s 3 ~/fictional_professions/detectives
```

In a pathname, the ~ represents the home directory.

If the file you specify does not exist, `mail` will create it. If the file does exist, `mail` will append the letter you're saving to the end of the file.

Saving a file removes it from your mailbox; `mail` displays an asterisk (`*`) next to the header of any letter that has been saved.

To leave the letter in your mailbox while appending it to a file, use the copy command:

```
& c 3 ~/fictional_professions/detectives
```

Saving and Copying Letters in Folders

You can dispense with typing full pathnames to files if you save or copy letters in mail *folders*. Folders are special files that are stored in a *folder directory*.

The advantages to saving or copying letters in folders is that your letters are automatically kept together in the same directory, where they are easily accessible without typing long pathnames.

Setting the Folder Directory

To use folders, you must first set up a folder directory. This is a two-step process:

- First, make the directory using mkdir.

  For example, if you wanted your folder directory to be called Letters, you would first make the directory:

```
venus% mkdir Letters
```

- Second, use a text editor or the SunView Defaults Editor to edit your .mailrc file (which contains mail options) to set the folder directory path. (Appendix B contains a detailed discussion of mail options and how to set them.) Here, you need to edit the ''set folder'' variable to include the pathname of your newly created folder directory:

```
set folder=/home/trouble/aphrodite/Letters
```

Or

```
set folder=~/Letters
```

Now your folder directory is set to receive letters saved in folders. (The change to the .mailrc file will take effect the next time you start mail.)

Designating Folders

You use the same commands to save or copy letters in folders as into files, except that the folder name is preceded by a + (plus sign) instead of a pathname. The + tells mail that the folder is to be kept in the folder directory (Letters).

For example, to save letter 3 in a folder called noir_writing, you would give the command:

```
& s 3 +noir_writing
```

mail interprets this command as meaning "save letter 3 into /home/trouble/aphrodite/Letters/noir_writing." (If the folder doesn't already exist, mail will create it.)

Copy the letter into a folder by giving the command:

```
& c 3 +noir_writing
```

Sending a Letter Directly to a File or Folder

You can send copies of your letters directly to one of your files or folders. To send a copy to a folder, simply type the folder name in either the Cc: or the Bcc: field. Sending a copy to file is similar, but you must include the full pathname.

Reading Letters in Files and Folders

To read letters saved in a file, use the command form

`mail -f` *filename*

Using the example above, you would read the file ~/fictional_professions/detectives by giving the command:

```
venus% mail -f ~/fictional_professions/detectives
```

You can read letters saved in a folder with a similar command—just use the + instead of a pathname. For example, to read the letters in the folder noir_writing, you would give the command:

```
venus% mail -f +noir_writing
```

Remember that + designates the folder directory.

The command starts mail in the file or folder designated. Only headers for the letters in the file or folder are displayed. Select a letter to read by typing its number at the mail prompt and pressing [Return].

You can also work on mail folders within the mail program. To see a list of your folders, type at a mail prompt

```
folders
```

To switch from your mailbox to a folder, use the command form

`folder +`*foldername*

To return to the mailbox, type at a mail prompt

```
%
```

To return to the previous folder, type

```
#
```

11.7. Using `vi` With `mail`

You can use the `vi` text editor to compose letters while running `mail`. This gives you the capability of correcting mistakes and adding and deleting text before you send your letters.

Under `mail` you can use the standard `vi` commands for inserting, deleting, and changing text. See Chapter 5 for a tutorial.

To write a letter with `vi`:

- Give the `mail` command with an address at either the `mail` prompt (`&`) or the command prompt (`venus%`).
- Type in the subject at the `Subject:` line. Press Return.
- Start `vi` by giving the command `~v` on a new line. The `vi` screen will appear, representing an empty file in your `/tmp` directory.
- Use `vi` commands to input and edit the body of your letter.
- When done, quit `vi` with the command `:wq`.

  After you quit `vi`, `mail` displays `(continue)`: here you can either add to the letter (outside `vi`) or send the letter by pressing Ctrl-D.

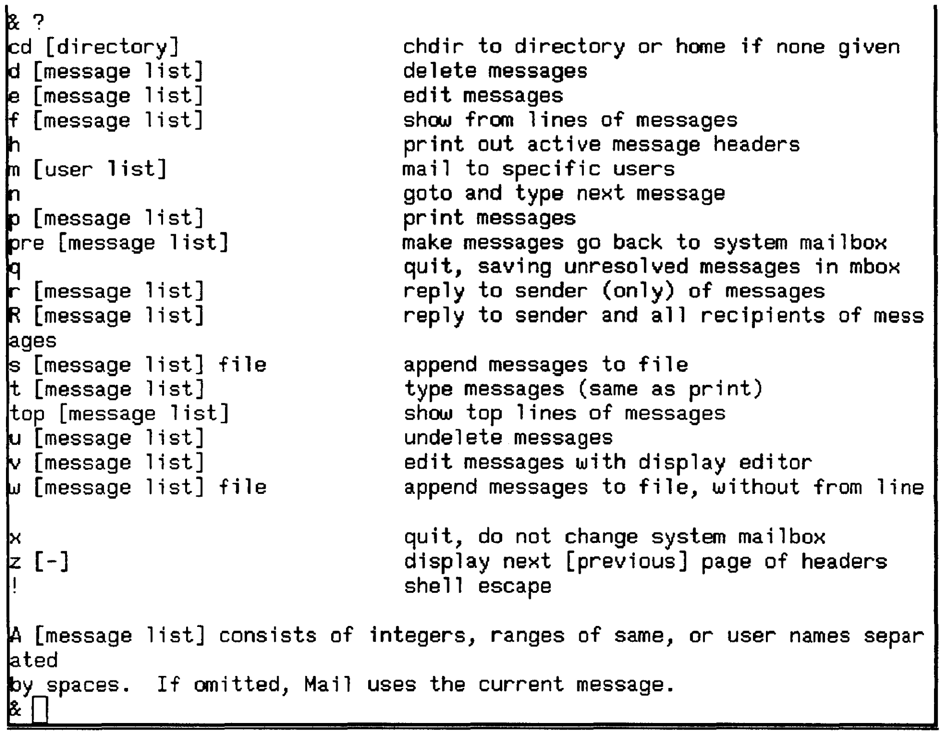
11.8. Getting Help: Other `mail` Commands

`mail` has two help commands that display lists of commands and functions. When in command mode, you can type `?` at the `mail` prompt (`&`) to see a list of commands used in that mode. Likewise, when in input mode (i.e., when writing a letter), you can give the equivalent command, `~?` to view a list of tilde commands (also called ''tilde escapes'').

As you become more familiar with the SunOS operating system and `mail`, you may want to try out some of these commands.

Give the commands below at the `mail` prompt (`&`):

Figure 11-1 *`mail` Prompt Commands*



```
& ?
cd [directory]                chdir to directory or home if none given
d [message list]              delete messages
e [message list]              edit messages
f [message list]              show from lines of messages
h                             print out active message headers
m [user list]                 mail to specific users
n                             goto and type next message
p [message list]              print messages
pre [message list]            make messages go back to system mailbox
q                             quit, saving unresolved messages in mbox
r [message list]              reply to sender (only) of messages
R [message list]              reply to sender and all recipients of mess
ages
s [message list] file         append messages to file
t [message list]              type messages (same as print)
top [message list]            show top lines of messages
u [message list]              undelete messages
v [message list]              edit messages with display editor
w [message list] file         append messages to file, without from line

x                             quit, do not change system mailbox
z [-]                         display next [previous] page of headers
!                             shell escape

A [message list] consists of integers, ranges of same, or user names separ
ated
by spaces.  If omitted, Mail uses the current message.
& 
```

The following commands are given only on new lines when composing or editing a letter:

Figure 11-2 *mail Tilde Commands*

```
& m marlowe@baycity
Subject: wanna help

~?
-------------------- ~ ESCAPES ----------------------------
~~              Quote a single tilde
~a,~A           Autograph (insert 'sign' variable)
~b users        Add users to Bcc list
~c users        Add users to Cc list
~d              Read in dead.letter file
~e              Edit the message buffer
~m messages     Read in messages, right-shifted by a tab
~f messages     Read in messages, do not right-shift
~h              Prompt for To list, Subject and Cc list
~p              Print the message buffer
~q,~Q           Quit, save letter in $HOME/dead.letter
~x              Quit, do not save letter
~r file         Read a file into the message buffer
~s subject      Set subject
~t users        Add users to To list
~v              Invoke display editor on message
~w file         Write message onto file
~.              End of input
~?              Print this message
~!command       Run a shell command
~|command       Pipe the message through the command
~:command       Execute regular Mail command
-----------------------------------------------------------
```

The man pages contain extensive information on `mail` in more technical form. To see this entry, give the command:

```
venus% man mail
```

12

Other Mail Features

There are several commands that can help you read and send mail and messages. This chapter introduces you to the use of

- The `from` command, which lets you find out who sent you mail;
- The `vacation` program, for responding to and forwarding mail when you're out of the office; and
- the `chfn` command, which allows you to change your name in the headers of mail you send.

12.1. Mail From Whom? The `from` Command

When you want to know whom your mail is from, without reading it using `mail` or Mail Tool, type `from` to your command prompt. For each letter waiting in your mailbox, `from` displays `From` followed by the sender's username, and the date and time it arrived in your mailbox:

```
venus% from
From sappho@aphrodite Sun Apr 1 8:45:12 1989
From rimbaud@verlaine Sun Apr 1 8:45:22 1989
From michaelangelo@david Sun Apr 1 8:45:45 1989
venus%
```

For more information on `from`, see the `from` man page, online or in the *SunOS Reference Manual.*

12.2. The `vacation` Program

You still receive mail when you're gone — even if your machine is turned off. The `vacation` utility automatically sends a pre-written response to anyone who sends you mail. Incoming mail is not affected; `vacation` acts like an electronic mail equivalent of a telephone answering machine.

Simply type `vacation` to start the program. It will help you create the file which contains the automatic reply. This file is called `.vacation.msg` and lives in your home directory. `vacation` automatically sets you up in your normal editor to edit a standard version of the reply letter.

You can modify `.vacation.msg` to say whatever you like. It should, however, start out with a `Subject:` line. If you include the word "$SUBJECT" in your reply, the subject of whatever letter you're replying to will be inserted at that point. Here's a sample `.vacation.msg` file:

Figure 12-1 *A Sample* `.vacation.msg` *File*

```
Subject:  I'm Away On Vacation.
Thanks for sending me your recent mail about $SUBJECT.
Currently I am in French Lick for the Indiana State
Free Throw Championships.  During this time, refer all
calls to my cat, Alfred.  I will be back on the 17th
of July.

Cosimo de Medici
```

This is how `vacation` works (in this case, using `vi`):

```
venus% vacation
This program can be used to answer your mail automatically
when you go away on vacation.
You need to create a message file in
/home/venus/medici/.vacation.msg first.
Please use your editor (/usr/local/vi) to edit this file.
  (Here you edit the sample vacation.msg)
You have a message file in /home/venus/medici/.vacation.msg.
Would you like to see it? n
Would you like to edit it? n
To enable the vacation feature a ".forward" file is created.
Would you like to enable the vacation feature? y
Vacation feature ENABLED. Please remember to turn it off when
you get back from vacation. Bon voyage.
venus%
```

To turn `vacation` off, or to modify your automatic reply letter, type `vacation` as you did to start it up.

`vacation` waits a specified interval before sending out your reply to someone it's already replied to; that way, someone who writes you several times while you're gone doesn't get your letter over and over again. (This specified interval is usually one week, but you can change it.)

Saving Mail With the `.forward` File

As shown above, `vacation` creates a file called `.forward`. This file is one line long and looks like this:

```
\user, "|/usr/ucb/vacation user"
```

Mail programs look in `.forward` to see where they should send mail addressed to you. In the case of the `.forward` file shown above, mail is sent to the user *user* and to the vacation program. You can modify `.forward` (which lives in your home directory) with an editor; you could, for example, forward copies of all your letters to another user or another machine. One of the most common ways people use `.forward` is to send copies of every letter they receive into a storage file. In the example below, the file `inbox` in the directory

/home/venus/medici gets a copy of all incoming mail. (Note that it doesn't send mail to vacation, although it could.):

```
venus% cat .forward
\medici, /home/venus/medici/inbox
venus%
```

NOTE *If you do forward mail to a file like* inbox, *be sure to prune it from time to time — it can get quite large!*

Complete information on vacation can be found either by reading the vacation section in the *SunOS Reference Manual* or by typing **man vacation**; information on the .forward file can be found by typing **man aliases**.

12.3. Changing Your Name With chfn

Another useful command to know is the chfn command. Below is an example of a typical letter header:

```
From finches@galapagos Sun Feb 12 17:02:36 1859
From finches@galapagos (Charles Darwin)
To: medici@venus
Subject:  Huxley's 'The Descent of Bulldog'
```

Note that the mail header not only has the username (*finches*) and machinename (*galapagos*) of the sender, it also give his real name (*Charles Darwin*). You can put your real name — or office extension, or job title, or whatever — in the header of messages you send by using the chfn command. (chfn stands for "change full name.") Or you can use the passwd command, which is introduced in *SunOS User's Guide: Getting Started* , with the -f option. The two commands are equivalent.

Here's how Charles Darwin would change the phrase "Charles Darwin" to "Charles Darwin, *famous biologist*":

```
galapagos: chfn
Changing finger information for finches on galapagos.
Default values are printed inside of '[]'.
To accept the default, type <return>.
To have a blank entry, type the word 'none'.

Name [Charles Darwin]: Charles Darwin, famous biologist
galapagos:
```

Now, messages from finches@galapagos will look like this:

```
From finches@galapagos Sun Feb 12 17:02:36 1859
From finches@galapagos (Charles Darwin, famous biologist)
To: medici@venus
Subject:  Huxley's 'The Descent of Bulldog'
```

CAUTION **Be careful *not* to use parentheses in your name field. Parentheses are automatically added by the mail system.**

13

Mail Over Networks

This chapter describes remote networks for the purpose of understanding how to send mail across them. For more information about networks in general, see Chapter 8.

There are many different kinds of networks, and each uses a different syntax for addressing letters. Some networks aren't connected to *your* network, making it impossible to exhange mail with their users.

13.1. What Networks Are Out There?

Before attempting to send mail to someone on a remote network, you must find out which network they're on.

UUCP and the Internet are the major networks.

The UUCP Network

UUCP is a program that allows machines to use telephones to transmit data. You send mail to other users by sending it through intermediate machines; each machine-to-machine pathway is unique. UUCP can be used to communicate with machines across the United States and throughout the world.

Sending Mail to People on the UUCP Network

To send mail to someone on the UUCP network, you must know the *network path*, or sequence of machines the letter must travel through to get from your machine to the recipient's machine.

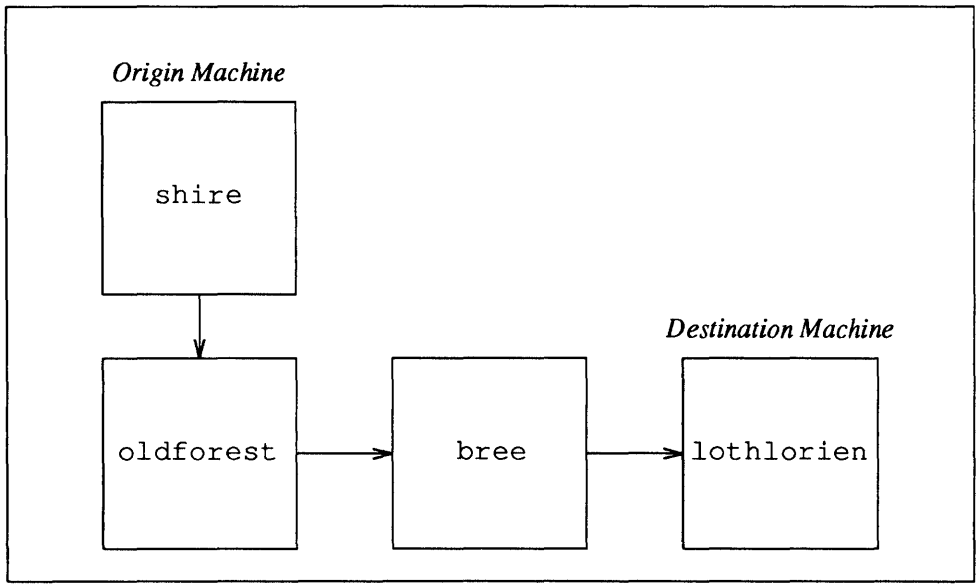
To find out machine name sequences necessary for mail addresses, ask prospective letter recipients if they know the appropriate network path. At the least, find out the prospective mail recipient's username and machine name.[5]

You can figure out the prospective recipient's mail address from this sequence of machine names. Pretend to walk along the path between the two machines starting with the first machine in the sequence and separating each *machine name* with an exclamation point (`!`), also called "bang." Add the recipient's *username* to the end of the address after one last exclamation point.

For example, to figure out the mail address that user `bilbo` on machine `shire` would use to send mail to user `galadriel` on machine `lothlorien`, walk from `shire` to `lothlorien`.

[5] When the letter recipient doesn't know the appropriate mail address, ask your system administrator, if you have one.

Figure 13-1 *Sequence of Machines in Network*



The sequence of machine names is: `oldforest`, `bree`, and `lothlorien`. The recipient's username is `galadriel`.[6] So the complete mail address is:

```
oldforest!bree!lothlorien!galadriel
```

When you specify the mail address on the command line after `mail`, make sure to put a backslash character (\) before each occurrence of an exclamation point (`oldforest\!bree\!lothlorien\!galadriel` in the above example), so that the shell interprets the address properly.[7] However, it is not necessary to use a backslash when you're *already in* `mail` or Mail Tool. Backslashes are only needed when you're typing in an address as part of a command line.

You can learn about aliasing a mail address to another character string in the `mail` man page, online or in the *SunOS Reference Manual.*

How Does Someone Send Mail to Me on the UUCP Network?

When people with accounts on a UUCP machine ask you how they can send mail to you, try to come up with the appropriate network path. Determine your username, your machine name, and other machines you know your machine talks to using UUCP. Determine the other person's username, machine name, and associated machines. Hopefully, you will discover an associated machine in common, so that you can identify a network path between you.[8]

[6] These names and places come from J.R.R. Tolkien's *Lord of the Rings.*

[7] The shell usually interprets exclamation points as part of the history mechanism. Putting a backslash before each exclamation point requires the shell to interpret the exclamation points as regular characters, rather than as special history mechanism characters. See Chapter 7 for more information about the history mechanism.

[8] Some sites support `uuname`, a program which lists the names of systems accessible by UUCP, and `uupath`, which gives UUCP paths between known machines.

For more information on the UUCP network, see your system administrator or look in *System and Network Administration.*

The Internet

To send mail to someone on the Internet, you must find out the username and machine name of the mail recipient, usually by asking the recipient. Unlike the UUCP network, however, you don't need to know the names of all the machines between your machine and the recipient's. The Internet takes care of that part automatically.

Construct the mail address by typing the recipient's *username*, followed by an at-sign character (@), the recipient's *machine name*, and his or her institution's full domain name, which should end in one of the following suffixes:

`.COM`
Asigned to commercial organizations.

`.EDU`
Assigned to educational institutions, chiefly universities.

`.ORG`
Assigned to nonprofit agencies.

`.GOV`
Assigned to government agencies.

`.MIL`
Assigned to military organizations.

Note that you do not have to capitalize the suffixes — you can say, for example, `.com` instead of `.COM`. So, for user `lumpy` on machine `geewhiz` at Extreme South-Eastern Rhode Island University (domain name: `exsoeari.edu`), the appropriate mail address would be:

```
lumpy@geewhiz.exsoeari.edu
```

Providing your username and domain name should be sufficient for someone on an Internet-linked machine to send you mail.[9]

For more information on the Internet, see Chapter 8 and *System and Network Administration.*

[9] Sometimes, the situation gets more complex. Contact your system administrator or look in *System and Network Administration.*

A

Mail Aliases

A mail *alias* is a selection of user names grouped under a single name.

Use aliases when you want to send mail to the same group of people over and over. For example, if you want to send mail from time to time to `joe@donkey`, `bill@whitehouse`, and `laura@smiley`, you can create an alias called `buddies`; each time you send mail to `buddies`, you send it to all three people.

There are two different ways to set up aliases. One is to set up an alias in your `.mailrc` file. The other way is to use the file `/etc/aliases`. These two kinds of aliases work differently, and you set them up differently. The table at the end of this chapter summarizes these differences.

A.1. Aliases in `.mailrc`

`.mailrc` is located in your home directory. It contains a number of settings that control the behavior of `mail` and Mail Tool. `.mailrc` is explained in detail in Appendix B.

To add an alias to `.mailrc`, type:

```
venus% vi + ~/.mailrc      (+ puts you at the file's end)
(edit .mailrc)
```

You don't have to use `vi`: any other text editor will do. But you cannot set up aliases with the SunView Defaults Editor.

Each alias takes up one line, with no carriage returns. (The line can *visually* "wrap around" onto another line.) Each alias should contain the following, separated by spaces:

- The word "alias,"
- The name of the alias (one word), and
- The people (username and machinename) in the alias, separated by spaces.

The example below shows two aliases. The first (`buddies`) has three people on it. The second (`chums`) has eight. (The names are wrapped around on the screen, splitting `ray@eastbay` in the middle).

Figure A-1 *A* `.mailrc` Alias

```
alias buddies joe@donkey bill@whitehouse laura@smiley
alias chums dickie@tucan janep@rydall tolouse@lautrec ray@e
astbay barton@bridge luigi@pasta mom@home sally@dance
```

To send mail to people on a `.mailrc` alias, just address it to the name of the alias. Do *not* include the name of your machine. For example, if you set up `buddies`, don't send to `buddies@`*your-machinename*. Just mail to `bud-dies`.

Some things to note about aliases in `.mailrc`:

- Aliases in `.mailrc` are *private*. That means that only you can use them. If another user tries to send to `buddies`, he will receive a "user unknown" error message.
- `.mailrc` aliases are automatically expanded, when the mail goes out, to show everyone on the alias. When you send to `buddies`, your mail arrives as though you had typed everyone's name as recipients. So everyone receiving the mail knows who else received it, although they cannot tell that you used an alias to send it.

  For example, if you sent this letter:

```
venus% mail buddies
Subject: Removal of Deceased Personnel

Employees are instructed to please not remove
personnel who may expire in front of their
computers.  Instead, please contact the Office
of Human Disposal and fill out form I-386/B.
```

Someone receiving the message would see this (note the `To:` line):

```
To: bill@whitehouse joe@donkey laura@smiley
Subject:  Removal of Deceased Personnel

Employees are instructed to please not remove
(rest of message)
```

Aliases in `/etc/aliases`

Aliases in the file `aliases` in the directory `/etc` work similarly to those in `.mailrc`, but there are important differences:

- Aliases in `/etc/aliases` are *public*. This means that if you set up an alias there called `lunchers`, anyone can send to `lunchers@`*your-machinename* and thus make use of the alias.

- `/etc/aliases` aliases are not expanded when mail goes out. If you send to `luncers@`*machinename*, that's how the mail will read when received. This means that recipients will know what the alias is, but not necesarily know who else is on it.

  For example, suppose you send out this message:

```
venus% mail chewers@venus
Subject:  New Restaurant

There's a new Indo-Japanese Thai place that's
opened on Garcia Ave.  It specializes in meat
by-products from around the globe, and the
Almost Chicken Kiev is exquisite.  So, in it's
own way, was the Inquisition, of course.

Anyone wanna go?
```

  This is what would appear to a recipient (note that the `To:` line remains unchanged):

```
To: chewers@venus
Subject:  New Restaurant

There's a new Indo-Japanese Thai place that's
(rest of message)
```

The format of `/etc/aliases` aliases is somewhat different from those found in `.mailrc`.

The format is as follows:

- the name of the alias, followed by a colon
- recipients (usernames and machinenames), separated by commas. The alias does not have to be on a single line

Here's how you'd set up two aliases, `lunchers` and `chewers`:

Figure A-2 `/etc/aliases` Aliases

```
lunchers: joe@donkey, bill@whitehouse, laura@smiley, tecun@uman,
          flann@swim2birds, odysseus@ithaka, weasel@toadhall
chewers:  bradk@zoohouse, jody@taxi, marmot@tomram, lunk@head
```

You must become `root` to modify your `/etc/aliases` file.[10] Here's how you would set up or modify an `/etc/aliases` alias (you can use another

[10] For more on becoming `root` or "superuser," see *SunOS User's Guide: Doing More.*

editor besides vi):

```
venus% su    (become root)
# vi + /etc/aliases    (+ puts you at the file's end)
(edit /etc/aliases)
# Ctrl-D (leave root)
venus%
```

NOTE *Aliases should always go at the end of* /etc/aliases.

When you send mail to an alias, be sure to add the name of the machine on which it's located. If someone has set up an alias called air_users on the machine canute, then you should send your mail to air_users@canute.

For more information, see "addresses" in the *SunOS Reference Manual*, or type man aliases or man addresses at a system prompt.

Table A-1 *Comparing Aliases:* .mailrc *and* /etc/aliases

	.mailrc	/etc/aliases
Must be root to modify?	no	yes
Send message to:	*alias*	*alias@machinename*
Recipients list seen by recipients?	yes	no
Names separated by commas?	no	yes
Names all on one line?	yes	no
Others can use?	no	yes

B

Modifying `mail` and Mail Tool

The `.mailrc` file contains a number of parameters that affect the way `mail` and Mail Tool work. By changing this file you can customize these programs to behave the way you want them to.

`.mailrc` is usually located in your home directory. It can be viewed with `more` and edited with `vi` or the SunView Text Editor. But if you're running SunView, you should use the Defaults Editor to modify `.mailrc` (except for the creation or deletion of aliases).

There is a sample `.mailrc` file, called `Mailrc`, in `/usr/lib`. It contains many convenient option settings; to get a copy for your use, copy it to your home directory as follows. (Before you do, type **`ls ~/.mailrc`** to see if you already have your own `.mailrc` file. If you get nothing back but your prompt, then you don't.)

```
venus% cp /usr/lib/Mailrc ~/.mailrc
venus%
```

This is what the default `Mailrc` file looks like:

Figure B-1 *The Default* `Mailrc` *File*

```
set alwaysignore
set askcc
set asksub
set autoprint
set cmd="lpr -p &"
set crt=15
set DEAD=~/dead.letter
set EDITOR=/usr/ucb/ex
set hold
set indentprefix=">"
set keepsave
set metoo
set PAGER="cat -s | more -22 -c"
set prompt="{Mail}&"
set record=~/.record
set SHELL=/bin/csh
set VISUAL=/usr/ucb/vi

ignore apparently-to date errors-to from id in-reply-to \
       message-id precedence received references remailed-date \
       remailed-from return-path sent-by status via
```

B.1. Using the Defaults Editor

A *default* is the automatic, assumed value of a setting; i.e., the value it has if you do nothing. If, for instance, you normally leave your office door open, you can say that the default for the door is "open."

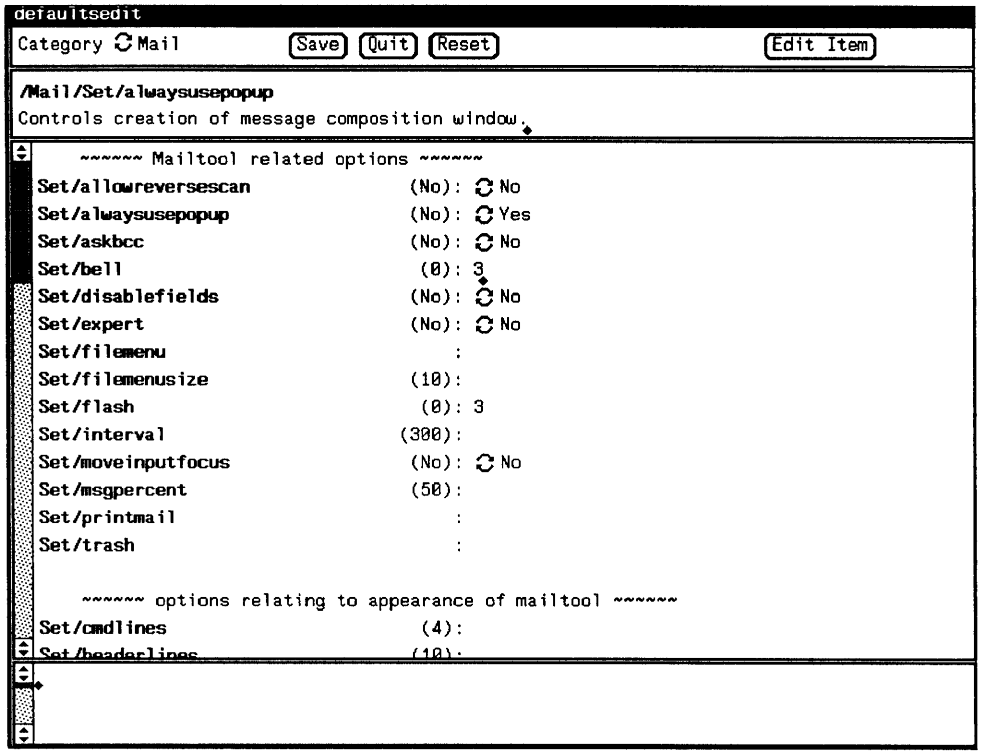
The Defaults Editor is a simple, interactive program that lets you change a number of SunView defaults in addition to those in `.mailrc`. The Defaults Editor presents you with a number of settings you can change, with short explanations of each item.

To bring it up, choose 'Defaults Editor' from the 'Editors' menu in SunView. Or you can give this command:

```
venus% defaultsedit &
venus%
```

The *SunView User's Guide* contains an explanation of how to use the Defaults Editor.

Figure B-2 `.mailrc` *in the Defaults Editor Window*



```
defaultsedit
Category Mail        [Save] [Quit] [Reset]                    [Edit Item]

/Mail/Set/alwaysusepopup
Controls creation of message composition window.

     ~~~~~~ Mailtool related options ~~~~~~
Set/allowreversescan               (No):  No
Set/alwaysusepopup                 (No):  Yes
Set/askbcc                         (No):  No
Set/bell                            (0):  3
Set/disablefields                  (No):  No
Set/expert                         (No):  No
Set/filemenu                           :
Set/filemenusize                   (10):
Set/flash                           (0):  3
Set/interval                      (300):
Set/moveinputfocus                 (No):  No
Set/msgpercent                     (50):
Set/printmail                          :
Set/trash                              :

     ~~~~~~ options relating to appearance of mailtool ~~~~~~
Set/cmdlines                        (4):
Set/headerlines                    (10):
```

With the Defaults Editor, many `mail` and Mail Tool options are turned on by setting them to 'Yes' and switched off by setting them to 'No.'

However, if you can't use the Defaults Editor, you can still modify `.mailrc` with a conventional editor such as `vi` or Text Editor. To set a `.mailrc` option, include a line like this:

```
set option
```

To turn an option off, either remove the above line or put the prefix *no* in front of the option's name:

```
set nooption
```

in `.mailrc`.

Some options in `.mailrc` are not simple "on/off" settings, but require you to put in a value. For example, to set your folder directory, you type its name in at the appropriate place in the Defaults Editor. If you're not using the Defaults Editor, you'd put in a line like

```
set folder=/home/medici/mail
```

(or whatever your folder directory will be).

NOTE *"Turning an option on," "setting an option," and using the Defaults Editor to set it to 'Yes' are all synonymous; "turning it off," "unsetting" it, and setting it to 'No' also all mean the same thing. But you can set a variable to 'No' only when you are using the Defaults Editor. Do not put a line in your* `.mailrc` *file that says* `set option=No`.

If you do not change a setting in `.mailrc`, then the default value is assumed.

For more information on `mail` and Mail Tool, and their associated options, see the *SunOS Reference Manual.* (Mail Tool is spelled *mailtool.*)

B.2. Mail Tool-Related Options

Scan mail in reverse with `allowreversescan`

When turned on, allows you to go through the letters in your mailbox in reverse order; i.e., last to first. This affects which letter is *next* — if the sense of direction is reversed, then the letter displayed by the **Next** button is actually the *previous* one. Default setting: turned off.

Get separate composition window with `alwaysusepopup`

Turning this on makes the composition window come up as a separate window frame; otherwise the composition window is simply split off from the message subwindow. Default: turned off.

Set automatic blind carbon copy prompt with `askbcc`

This gives a "Bcc:" prompt when set. "Bcc" stands for *blind carbon copy.* Like "Cc:", except that the list of people you copy the letter to doesn't appear in the letter header, so you can copy a letter to someone without alerting the addressee. Default: turned off.

Mail Tool beeps when mail arrives with `bell`

Number of times you want the terminal to beep when you get a letter. Default: no beep. See also `flash`.

Remove address fields with `disablefields`

Including this option removes the fields in the composition window's address template. Default: turned off.

Confirm edits in message window with `editmessagewindow`

With `editmessagewindow` turned on, if you try to edit a letter in the message window, Mail Tool will first ask you to confirm that you want to edit it. Default: turned off.

Disable confirmations with `expert`

When you have `expert` set, Mail Tool does not ask you to confirm deletions, cancellations, etc. Default: turned off.

Set File: menu with `filemenu`

The `File:` prompt in the command panel window has a menu of folders you've been working with. You can set `filemenu` so that certain files are automatically included on this menu when Mail Tool starts. Examples include +trash and +mbox. (See trash and MBOX, below.)

When you set a variable to have a value that is expressed by more than one word, put the variable in quotes. For example, if you want filemenu to include more than one file, put the list of files in quotes: '+trash +mbox'.

Set number of files in File: menu with filemenusize

Maximum number of files in the File: prompt's menu. Default: 10.

Mail Tool flashes when mail arrives with flash

Number of times to flash Mail Tool when mail arrives. Also flashes the Mail Tool icon when Mail Tool is closed. Default: 0.

Mail Tool cannot flash without beeping, but it can beep without flashing. This means that if you set flash to 3 and bell to 1, you will get one flash and one beep. If you set bell to 3 and flash to 1, you will get three beeps and one flash. See bell, above.

Change frequency new mail is checked for with interval

Time (in seconds) that Mail Tool waits before checking for new mail Default: 300 seconds (5 minutes).

Set prompt in composition window with moveinputfocus

moveinputfocus controls where you type when you start a composition window: with moveinputfocus set, the composition window automatically becomes the window you're typing in as soon as it comes up. This feature only has meaning if you are using "Click-to-Type," described in the *SunView User's Guide*. Default: turned off.

Change proportion of message to composition window with msgpercent

This controls how much of the message subwindow will be used for a composition window when composing a letter. Default: 50 percent. But if alwaysusepopup is set, this setting has no meaning.

Set print command with printmail

The command for printing a letter. You can use whatever printing scheme works best for you. Normally set to 'lpr -p'.

Store deleted letters with trash

trash is a file that collects your deleted letters; they stay here until you push the **Done** button. Setting trash allows you to look at deleted letters as though they were saved in a regular file. You set trash to the name of your trash file. If set to +trash, it can be accessed like any other folder.

Options Relating to the Appearance of Mail Tool:

Change size of header list window with headerlines

Default: 10 lines.

Change size of message window with `maillines`

Default: 30 lines.

Change size of composition windows with `popuplines`

Default: 30 lines.

B.3. Options Affecting Both `mail` and Mail Tool

Treat network names with matching usernames as identical with `allnet`

All network names whose usernames (e.g., `helen` in `helen@troy`, `helen@keller`, and `helen@lavosh`) match are treated as identical. Default: turned off.

Ignore Ctrl-C when typing letters with `alwaysignore`

See Section B.5 for more on `alwaysignore`.

Add letters to beginning of mbox with `append`

Normally letters are added to the end of `mbox`; if you prefer to have your most recent arrivals go to the beginning, set `noappend` or set to 'No' with the Defaults Editor. See `MBOX` and `hold`.

`ask`

No longer implemented. See `asksub`.

Set automatic carbon copy prompt with `askcc`

When this is set you are automatically given the "Cc:" (carbon copy) prompt when composing a letter. The default setting is to have this feature turned off.

Set automatic subject prompt with `asksub`

Automatically prompts for a subject when you're composing a letter. Default: turned on.

Display next letter after a deletion with `autoprint`

With `autoprint` on, `mail` and Mail Tool display the next letter in the mailbox when one is deleted. Default: turned off.

Specify file to store interrupted letters with `DEAD`

This ghoulish variable takes the name of a file (with its full path name) where partial letters get stored in case of an interruption like a power failure. `DEAD` is normally set to be a file called `dead.letter` in your home directory. The `save` variable must be set for this variable to take effect. See `save`.

Designate folder directory with `folder`

This is the directory that contains your mail folders. For more information, see the `outfolder` variable below.

Keep letters in mailbox with `hold`

When hold is turned on, letters you've read are still kept in your mailbox until you save or delete them. When hold is turned off, already-read letters are moved to a file (usually located in your home directory and called mbox) when you do a commit. hold is turned off for mail, on for Mail Tool. See MBOX.

Indent included letters with indentprefix

When composing a letter, you can include another letter, indented to set it off. indentprefix is what gets put to the left of a letter when it's indented. The default is just a tab; you can put in one or more characters of your choice, surrounded by quotes, to indicate that this is an included letter. A tab is indicated by a Ctrl-I or ^I.

Keep mailbox with keep

keep signals that you want to keep your mailbox even when it's empty. Turning keep on means that your mailbox is truncated to zero length when empty, instead of removed and created anew when you get mail. Default: turned off.

Retain saved letters in mailbox with keepsave

Normally when you save a letter into a file or folder, you delete it from your mailbox. (In Mail Tool, letters are not deleted if you use the 'Copy' option of Save.) Setting keepsave prevents mail or Mail Tool from deleting it automatically.

Remember, multi-word variable values, like 'ls -F', should go in quotes.

Set command to display folder directory with LISTER

LISTER is set to a SunOS shell command which you use for displaying the contents of your folder directory. In Mail Tool, the default is ls -F; in mail, the default is ls.

If you change LISTER, the command you replace it with must display directories as a "/" the way ls -F does, for Mail Tool to work correctly. See ls in the *SunOS Reference Manual*or type **man ls** for more information.

Designate mailbox file with MBOX

Normally, letters you've read are kept in your mailbox until deleted or saved. When hold is turned off, however, these letters are saved into a file specified by MBOX. Normally this is a file called mbox in your home directory. See hold.

Designate yourself as recipient with metoo

When you send something to an alias group of which you're a member, you don't receive a copy of the letter unless you specifically address it to yourself as well. Setting metoo, however, includes you among the recipients. Default: turned off. metoo will work for alias groups that you declare in your .mailrc file. Whether it works for other mail aliases depends on how your system is set up.

Addresses given relative to yours with `onehop`

When you receive a letter that was sent to several people, the other recipients' machine addresses are normally given relative to the *author's* address. Setting onehop allows the others' addresses to be given relative to your own — i.e., just "one hop" away from you, not through the author. This makes your replies more efficient.

Place letter record in folder directory with `outfolder`

You can keep a record of every letter you send; they go into a file set by the variable `record`. If `outfolder` is turned on, then this file will be located in your folder directory. Default: turned off. See also `folder` and `record`.

Designate letter record file with `record`

You set `record` to be the name of a file that receives a copy of every letter you send. If `outfolder` is turned on, then this file is located in your folder directory (set with the `folder` variable). If `outfolder` isn't turned on, then `record` should include the full pathname of this file.

Reverse reply commands with `replyall`

The net effect of setting `replyall` is to reverse `mail`'s **R** and **r** commands or, in Mail Tool, to reverse the meanings of 'Reply' and 'Reply (all).' Default: turned off.

Store interrupted letters with `save`

When turned on, `mail` and Mail Tool save partial letters into the file specified by `DEAD` in case of an interruption like a power failure. Default: turned on. See `DEAD`.

Specify mail delivery program with `sendmail`

`mail` and Mail Tool usually use the program `sendmail` to deliver mail; you can specify an alternate program here.

Display letter recipient with `showto`

Sometimes you send letters to yourself (for example when you "Cc:" yourself or send a letter to an alias group that includes you). If you set `showto`, `mail` and Mail Tool display the letter's recipient, rather than the sender (who is yourself). That way you see why you received mail you sent, rather than that you sent it. By default, `mail` and Mail Tool display the sender in all cases.

B.4. Options That Affect Only Mail

`bang` (for advanced users)

Enables the special-casing of exclamation points (!) in shell escape command lines as in `vi`. ! (not followed by a shell command) gives you a history of shell commands you've done in mail. Default: turned off.

Example: `set cmd='lpr -h'`. When you type | ("pipe") at the mail prompt, it is interpreted as "Pipe to lpr -h."

`cmd` (for advanced users)

The default shell command for the `pipe` command in `mail`.

Convert uucp addresses to internet with `conv`

Convert uucp addresses to the specified address style. The only valid conversion now is `internet`, which requires a mail delivery program conforming to the RFC822 standard for electronic mail addressing. Conversion is normally disabled. See the `-U` command-line option in the `mail` section of the *SunOS Reference Manual.*

Set crt number of lines with `crt`

`crt` is a number roughly corresponding to the number of vertical lines in many terminal screens. If a letter has more than this number of lines, `mail` pipes it through a displaying command set by `PAGER` (usually the `more` command). Default: turned off. See `PAGER`.

Terminate letters with a `dot`

Accept a dot ('.') alone on a line to terminate a letter. This is the default. The variable `ignoreeof` takes precedence over dot. See `ignoreeof` in this section.

Edit mail headers with `editheaders`

If set, the message headers are included in the text to be edited by the `~e` and `~v` commands.

Set editor with `EDITOR`

The `edit` and `~e` commands invoke an editor to use when writing letters; `EDITOR` sets the editor to use. The default is `ex`.

Change mail escape character with `escape`

You can replace the `~` in commands such as `~h`, `~e`, and `~m` with a character of your own choosing.

Turn off header list display with `header`

This variable is normally set, so that when you enter `mail` the header list is displayed. You can suppress the initial display of the header list by turning this variable off.

Ignore [Ctrl-C] **when typing letters with** `ignore`

You may wish to ignore a [Ctrl-C] when typing a letter, especially if you have a noisy dial-up line.

Terminate letters with dot or tilde-dot with `ignoreeof`

Normally you terminate a letter with a [Ctrl-D]. Setting `ignoreeof` means you must end a letter with either a period on a line by itself or the `~.` command. `ignoreeof` takes precedence over `dot`.

Insert form feeds with `page`

Insert a form feed after each letter. Default: turned off.

Set command to display long letters with PAGER

The command that mail uses to display long letters— usually more. See crt.

Set mail prompt with prompt

Set to & by default. You can change the symbol.

Suppress display of first letter with quiet

mail normally displays a short message, including its version number, when it starts up. You can turn this off to suppress this message.

Set number of header summaries displayed with screen

The maximum number of header summaries to be displayed at one time; i.e., the amount of the screen to be taken up by the header list. No default.

Mail waits to return prompt with sendwait

With sendwait set, mail (or Mail Tool) waits until it has finished sending off a letter before coming back to the user. Default: turned off.

Set command interpreter with SHELL

The name of a preferred command interpreter; usually sh; you can set this to (for example) /bin/csh. The command interpreter is inherited from the environment unless you specify it here.

Remember to enclose your signature in quotes.

Designate special signature with sign

You can "sign" your letter with the ~a command in mail; the sign variable is your "signature." It could be some pithy phrase you want to finish your letters with.

Specify number of lines displayed with toplines

The top command in mail prints out the first few lines of letters whose letter numbers you give. toplines specifies how many lines to print out. Default: 5 lines.

Set verbose option with verbose

When set, sendmail is used with the -v (verbose) option. Default: not set. See sendmail (8) in the *SunOS Reference Manual* or type **man 8 sendmail**.

Specify screen editor with VISUAL

The name of the screen editor used when you type the ~v command in mail. Default: vi.

B.5. Suppressing Header Lines in Letters

Your letters begin with a header that contains lines like these:

```
From loeb@leopold Fri Aug 21 15:18:42 1924
Return-Path: <loeb@leopold>
Received: from leopold.XXX.uucp by darrow.XXX.uucp (3.2/SMI-3.0)
    id AA21411; Fri, 21 Aug 87 15:18:32 PDT
Received: by leopold.XXX.uucp (3.2/SMI-3.0)
    id AA20410; Fri, 21 Aug 24 15:22:11 PDT
Date: Fri, 21 Aug 24 15:22:11 PDT
From: loeb@leopold (Arthur Garfield Hayes)
Message-Id: <8788212322.AA00410@leopold.XXX.uucp>
To: clarence@darrow
Subject: Scopes Case Background
Status: RO
```

You can have `mail` and Mail Tool suppress the display of any of these lines by including an "ignore" line in your `.mailrc` file. For example, the following line

```
ignore message-id return-path received date status via
```

produces a header that looks like this:

```
From loeb@leopold Fri Aug 21 15:18:42 1924
From: loeb@leopold (Arthur Garfield Hayes)
To: clarence@darrow
Subject: Scopes Case Background
```

You can pick and choose which of the header categories you want to have displayed.

You cannot use the Defaults Editor to add an ignore line to `.mailrc`. You must edit `.mailrc` with a conventional editor such as `vi` or Text Editor.

The header categories that you tell `mail` and Mail Tool not to display are still included when the letter is saved. However, if `alwaysignore` is set, then these header lines are not saved, either. This is also true for copying and including letters as well.

In Mail Tool, you can use the **Show** button's menu to override the `ignore` option. The 'Show Full Header' option will display a letter's full header.

Index

Notes

Notes

Notes

Notes

Notes

Systems for Open Computing™

Corporate Headquarters
Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 37-29639

For U.S. Sales Office locations call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Bagshot Manor, Green Lane
Bagshot, Surrey GU19 5NL
England
0276 51440
TLX 859017

Australia: (02) 413 2666
Canada: 416 477-6745
France: (1) 40 94 80 00

Germany: (089) 95094-0
Hong Kong: 852 5-8651688
Italy: (39) 6056337
Japan: (03) 221-7021
Korea: 2-7802255
New Zealand: (04) 499 2344
Nordic Countries: +46 (0)8 7647810
PRC: 1-8315568
Singapore: 224 3388
Spain: (1) 2532003
Switzerland: (1) 8289555
The Netherlands: 033 501234

Taiwan: 2-7213257
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276 51440

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales

sun
microsystems

Part Number: 800-3830-10
Revision A, of March 27, 1990