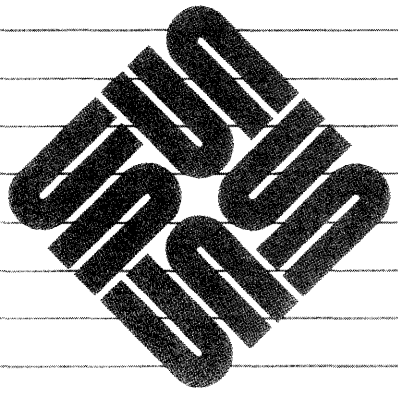
sun®
microsystems

Getting Started *with* UNIX: Beginner's Guide



Part No: 800-1274-03
Revision A of 17 February 1986

Credits and Trademarks

Sun Workstation® is a registered trademark of Sun Microsystems, Inc.

SunStation™, Sun Microsystems™, SunCore™, SunWindows™, DVMA™, and the combination of Sun with a numeric suffix are trademarks of Sun Microsystems, Inc.

UNIX, UNIX/32V, UNIX System III, and UNIX System V are trademarks of AT&T Bell Laboratories.

Intel® and Multibus® are registered trademarks of Intel Corporation.

DEC®, PDP®, VT®, and VAX® are registered trademarks of Digital Equipment Corporation.

Contents

Tables

Figures

Preface

Welcome to the Sun Workstation. This document introduces the basic tools you need to use the system. We assume that you are new to the Sun Workstation, and have little or no experience with UNIX† or similar operating systems. The information we present applies particularly to Sun products; most of the information also applies to anyone using UNIX.

We provide exercises to learn the system, not detailed explanations of the inner workings of commands and programs. In each of the *Beginner's Guides*, we refer you to the other Sun documentation, drawing a road map for you to follow when you wish to learn more about a certain topic.

Getting Started with UNIX describes the workstation, the commands you will use most frequently, the keyboard, the terminal screen, and the UNIX system file structure. In addition, this manual presents simple editing, formatting, and searching programs, and includes special sections that give timesaving features, online documentation, where to learn more, a command summary, a glossary, an annotated bibliography, and a quick reference.

Companion documents

Setting Up Your UNIX Environment: Beginner's Guide
Windows and Window-Based Tools: Beginner's Guide
Mail and Messages: Beginner's Guide
Self Help With Problems: Beginner's Guide
Doing More With UNIX: Beginner's Guide

Formatting Documents on the Sun Workstation
Editing Text Files on the Sun Workstation
Commands Reference Manual

† UNIX is a trademark of AT&T Bell Laboratories.

1

A Work Session

1

A Work Session

Before you start your first UNIX† work session, you will need to make sure that your system hardware is set up, that you have an account on your system, and that, when possible, you can ask a regular user of the system for help.[1]

This chapter will help you learn how to:

- log in to the machine
- set and change your password, and
- log out

Turn on the workstation (the switch is on the back) and sit in front of it while you read this tutorial, so you can try the examples, and so you can remember them later on. If something does not work exactly the way you expect, and it still won't work after you make a few good guesses, then ask a regular user of the system for help. Or, look in *Self Help With Problems: Beginner's Guide*.

Don't be afraid to play with the computer — it won't bite you. The best way to learn is to dig in confidently and try some examples.

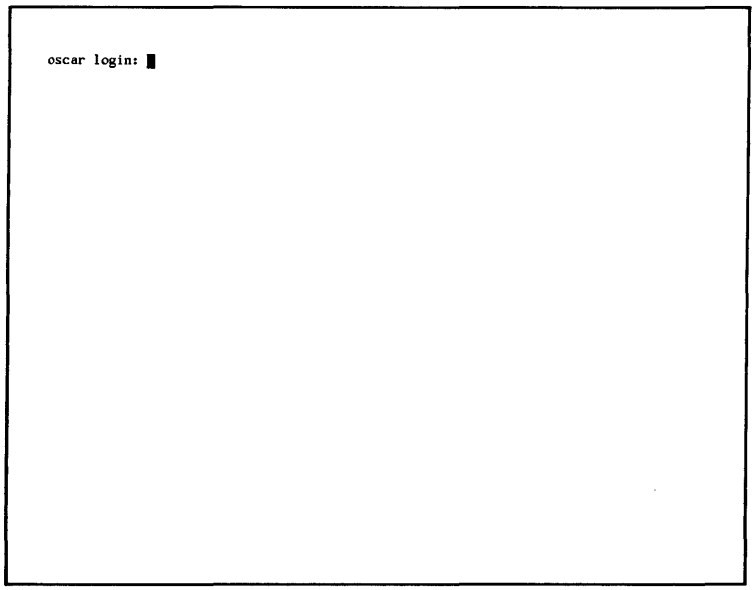
† UNIX is a trademark of AT&T Bell Laboratories.

[1] Hopefully, your system administrator will have arranged this for you. If not, or if you are the system administrator, consult the *System Administration for the Sun Workstation* manual to set up the system and accounts before reading this manual.

1.1. Log In

On your workstation screen, you should see something like this:

Figure 1-1 *Screen With Login Prompt*



This is called the *system login prompt.* In most cases, the word before `login:` is the name of your machine; for this example, the machine name, or *hostname*, is `venus`.

Note: UNIX is an *operating system*, in other words, a collection of programs that monitors the use of the machine and supervises the other programs executed by it.

The UNIX operating system will allow more than one person to use the system at the same time, so 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 the section on changing your password, Section 1.2 .

Note: If you make a mistake in typing a character of your username or password, type the DEL key to *delete*, or erase, the character, then type the correct character in place of the incorrect one.

At the login prompt, type your username, followed by the carriage return key RETURN, (for the purposes of instruction in this manual, the bold-faced characters represent what you type to the system; the system types the characters that are the regular font intensity):

Figure 1-2 *The Password Prompt*

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

The username is `medici` in this example. Be aware that the Sun system is *case-sensitive* — it distinguishes between lower-case and upper-case characters — and has a strong orientation toward *lower-case characters*.

Note: Watch out for the CAPS LOCK key, if there is one on your keyboard. If you should log in using upper-case for your username or password, see the section on logging out, Section 1.3 to log out and try logging in again.

The system will not do anything until you complete the command by typing a carriage return (the RETURN key). After it accepts your username (it may "think" for a second or two), it will prompt you for your password.[2]

At the password prompt, type your password.

As you can see, the system will not *echo* (type out) your password on the screen. This is so that other users who are peeking over your shoulder will not discover your password.

Don't forget to type RETURN after your password, so the system will interpret what you typed. From now on, you'll have to type RETURN after each command without a reminder from this manual.

When you mistype your username or your password, the system will respond like this:

Figure 1-3 *Incorrect Login*

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

Simply retype your username and password to log in.[3]

After you correctly type your username and password, the system will pause for less than a minute, then type out something like this:

Figure 1-4 *System Login Information*

```
venus login: medici
Password:
Last login: Fri Oct 31 23:59:59 from console
Sun UNIX 4.2 Release 3.0 (DIONE_CLIENT) #1: Fri Feb 14 00:00:01 PST 1986
venus%
```

In this example, the first line the system types tells when your username last logged in to the system (`console` refers to a login while sitting at the workstation keyboard). When you log in, you may want to check this line to see if it confirms your recollection of when you last logged in, so you can be aware if

[2] If the account does not have a password assigned to it, the system will log you in without asking for a password.

[3] 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 Administration for the Sun Workstation* manual.

someone else is using your account. The second line, with the word `UNIX` in it, tells some information about the version of the system that is running. Don't worry about this for now.[4]

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 of 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 more information on how to read your mail, see *Mail and Messages: Beginner's Guide.*

The last line in this example is the *command prompt*, consisting of your machine name, `venus` in these examples, and a percent sign (`%`).[5]

Congratulations! At this point, you have successfully logged in to the Sun Workstation.

1.2. Change Your Password

Right after you log in for the first time, change your password, so you are the only one who has easy access to your account. Change your password immediately if you believe someone has used your account without your permission. You may want to change your password periodically for security reasons.

Note: One good way to remember a password is to think of a word that is the answer to a provocative question.

Your personal password is your choice entirely. Pick a password that you can remember without writing it down. For convenience, you may want to pick a password that is easy to type.

---

[4] If you have ever logged in to your workstation from another system, you will see something like this:

```
venus login: medici
Password:
Last login: Fri Oct 31 23:59:59 from pandora
Sun UNIX 4.2 Release 3.0 (DIONE_CLIENT) #1: Fri Feb 14 00:00:01 PST 1986
venus%
```

As before, one line of system information lets you know the last time your username accessed the account, but this time it specifies from which machine (`pandora`) you last logged in.

[5] In some cases the command prompt may be just a percent sign (`%`), or just a dollar sign (`$`). To customize your command prompt, consult *Setting Up Your UNIX Environment: Beginner's Guide* for information about the `prompt` environment variable.

Then, type the command `passwd` as follows (if no password was assigned to your account, the system will skip the `Old Password:` prompt):

Figure 1-5 *Changing Your Password*

```
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. Just as during log in, the system will not display any passwords typed at password prompts. Then, type in the new password you picked when you see the next system prompt:

```
New password:
```

Because the password does not echo, the system will ask you to retype it to prevent typographical accidents.

Again, if the system doesn't have a password for you, it won't prompt you for

```
Old password:
```

it will skip directly to the

```
New password:
```

prompt, where you should enter the password you have picked.

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

```
Sorry
```

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

On most systems, when you pick a password less than six characters long, the system will not allow you to enter it unless you are persistent. Keep on typing in the new password even though the system says:

```
Please use a longer password.
```

until the system agrees to make the change.

1.3. Log Out

After completing your work, it is important to *log out*, or stop the current session of your account's access to the system. Anyone who walks by your workstation or terminal will have access to the system using your account, unless you remember to log out.

When you finish your work session, try to log out using the `logout` command:

Figure 1-6 *Logging Out With* `logout`

```
venus% logout

venus login:
```

Another way to log out is to type CTRL-D, the *end-of-file character* (explained further in the section of Chapter 4 on creating a file). To type a control character, hold down the CTRL key while typing the letter key. In this example, hold down CTRL and type the letter d. Even though CTRL-D specifies an upper-case letter `D`, you can type `d` without the SHIFT key, in other words lower-case `d`, in conjunction with CTRL.

This is what it looks like:

Figure 1-7 *Logging Out With* `CTRL-D`

```
venus% ^D

venus login:
```

Note: Turning off the workstation will not necessarily log you out. Most Sun systems do not have a "time-out" mechanism, so unless you log out explicitly, most often you will remain logged in to the system.

After you log out successfully, the system will display the login prompt again for the next user to log in.

[5] You may wish to prevent CTRL-D from logging you out. See the description of `ignoreeof` in *Setting Up Your UNIX Environment: Beginner's Guide* for information on how to do this.

When you encounter one of the following problems, respond as indicated in the table:

Table 1-1 *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`

I'm just completely stuck. (This usually works when logged into a machine over the network.)	Try typing: RETURN ~ . RETURN

Try logging out and logging in again for practice.

Congratulations! You've completed your first work session on a Sun.

2

The Keyboard

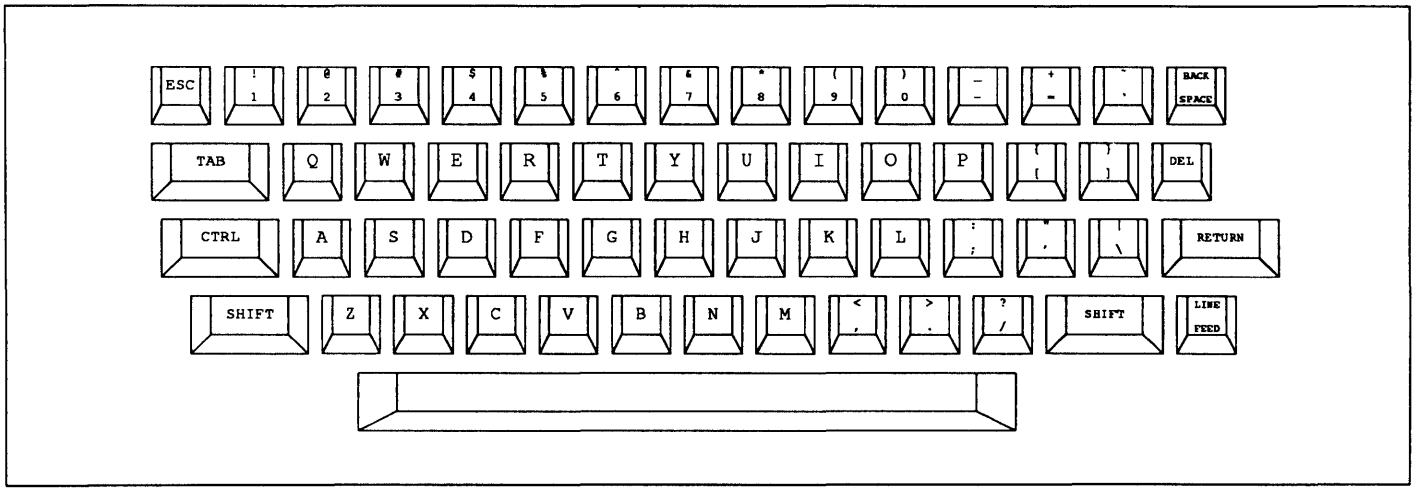
2

The Keyboard

2.1. Keyboard Layout

You can type commands to the system with the *keyboard.* The keyboard of a Sun workstation is like the keyboard of a standard typewriter, with the addition of some *special keys.* With your first work session, in Chapter 1 , you became acquainted with some of them. Now, you'll review those keys and learn about a few more.

Figure 2-1 *Standard Sun Workstation Keyboard*



Note: The system distinguishes between the one key, [1] and the letter `l` key, [l] It also distinguishes between the zero key, [0] and the letter `o` key, [O]

To type most keyboard characters, simply find the key labeled with the appropriate character and press the key until the character appears on the screen, or the desired action takes place. Be careful not to press the key for too long because many keys will repeat their function, varying with the length of time you press the key.

Remember that UNIX distinguishes between lower-case and upper-case characters and, in general, prefers lower-case.

2.2. The Carriage Return Key

One of the keys you encountered in Chapter 1 is [RETURN] the *carriage return* key. Located on the far right of the keyboard, [RETURN] causes the system to interpret a command line, or line that you are typing on, and execute it. Until you type [RETURN] you can correct the command line.[6]

2.3. The Space Bar and the Tab Key

Use the space bar, located along the front of the keyboard, to add spaces to the command line when necessary.

The tab key, [TAB] acts like the space bar, but usually inserts eight spaces, instead of one. [TAB] is located on the upper left of the keyboard.

2.4. The Delete and Back Space Keys

On each system, either the delete key, [DEL] or the backspace key, [BACK SPACE] allows you to back up one character and *rubout*, or erase, a specific character in the commands you type. To correct typing mistakes, you rubout as many characters as you wish, then retype these characters correctly. [DEL] and [BACK SPACE] are located near each other on the far right of the keyboard.

Note: You can only rubout characters back to the beginning of a command line.

Some systems use [DEL] as the rubout key; others use [BACK SPACE].[7] The best way to figure it out on your own system is to try it. For example, when you want to use the `passwd` command to change your password, but you type `paaswd` by accident, then press the [DEL] key four times, so that only `pa` shows, and retype the remaining characters, completing the correction. When your default rubout key is [DEL], it looks like this:

Figure 2-2 *Use of Delete Key*

Before:

```
venus% paaswd
```

During:

```
venus% pa
```

After:

```
venus% passwd
Changing password for medici on venus
...
```

[6] Learn more about commands and command lines in Chapter 5 .

[7] If you wish to change your default rubout key from [DEL] to [BACK SPACE], or *vice versa*, see *Setting Up Your UNIX Environment: Beginner's Guide.*

If instead, your default rubout key is [BACK SPACE], and you type [DEL], you see this:

Figure 2-3 *Use of Delete Key When Back Space Key is Default*

Before:

```
venus% paaswd
```

During:

```
venus% pa^?^?^?^?
```

Conversely, if [DEL] is the default rubout key on your system, then when you type [BACK SPACE], you will see something like this:

Figure 2-4 *Use of Back Space Key When Delete Key is Default*

Before:

```
venus% paaswd
```

During:

```
venus% pa^H^H^H^H
```

Try "mistyping" and correcting a few commands using the key that acts as the rubout key on your system.

2.5. Control Keys

Control keys are keys that require you to press [CTRL] and hold it down while typing another key. In this way, the keys can have two functions, one when pressed alone, and another when pressed with [CTRL]. This capability expands the usefulness of the keyboard, similar to the way the [SHIFT] key allows you to type both lower-case and upper-case characters.

The *line kill character*, CTRL-U, erases the entire command line. For instance, to erase the entire "mistyped" `passwd` command, press the CTRL key, and while you hold it down, type the u key. It looks like this:

Figure 2-5 *The Line Kill Character*

Before:

```
venus% pas sdw
```

After:

```
venus%
```

Notice that CTRL-U doesn't appear on the screen when you type it. Some control keys appear on the screen; others don't.

For now, don't worry about understanding all about control keys. But, here are some useful control keys:

Table 2-1 *Control Keys*

Key	*Appearance*	*Function*
CTRL-U	^U	Erases entire command line; the *kill character*
CTRL-W	^W	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

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

2.6. Escape Keys or Meta-Keys

Another way to expand keyboard functionality is the use of *escape keys* or *meta-keys*. In contrast to control keys, escape keys require that you type the ESC key, *release* it, then type the complementing key.

For example, to type ESC-U, press the ESC key, release it, then type the u key.[8]

Beginners do not often use escape keys, so you only need to know about them in passing.

[8] Some manuals refer to escape keys with any of these notations:
`<ESCAPE>-U` `<ESC>-U` `M-U` `<META>-U` `META-U`

3

The Terminal Screen and Mouse

3

The Terminal Screen and Mouse

When you sit down in front of your Sun Workstation, you probably see something like this:[9]

Figure 3-1 *Parts of the Sun Workstation*

It is possible that you are not using a workstation, but a terminal. In this case, you won't have a mouse or a mouse tablet, like in the illustration above.

In the last chapter (Chapter 2), you experimented with the keyboard. All along, you've used the *terminal screen* to view your work. Read on for descriptions of the other parts of the workstation.

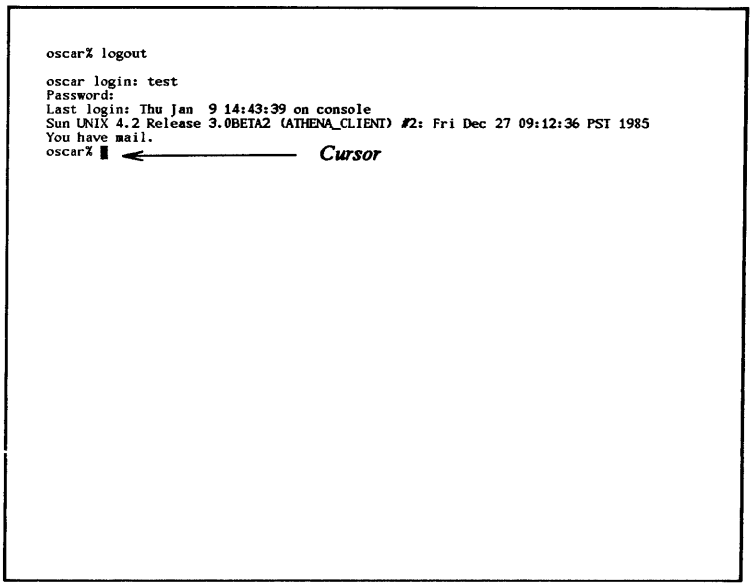
[9] If you are left-handed, you may wish to put the mouse and the mouse tablet on the left side, not the right side, of the workstation.

3.1. The Cursor

The *cursor* is the rectangular section on the screen that moves when you type. Notice that the keys you type appear in place of the cursor, and as soon as they appear, the cursor moves to the next character position on the screen.

Here is a diagram of the workstation screen labeled so as to point out the cursor:

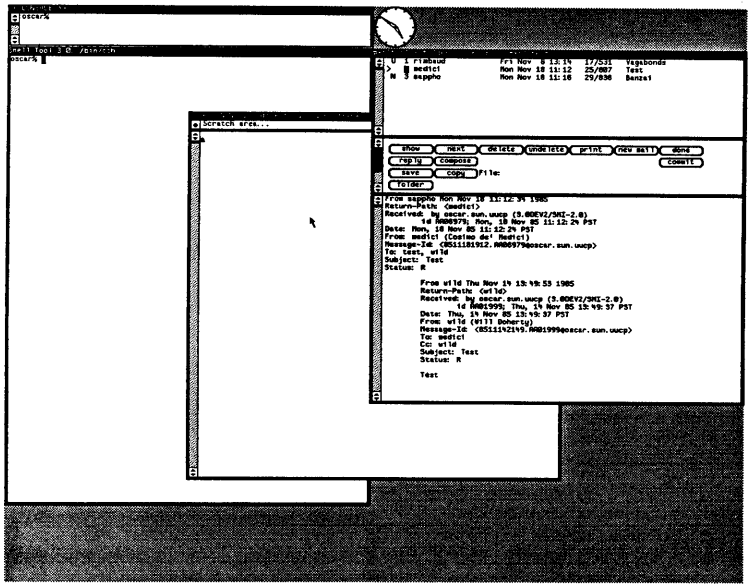
Figure 3-2 *The Cursor*



3.2. Windows

Your workstation screen may look something like this when you first log in:

Figure 3-3 *A Window System in Operation*



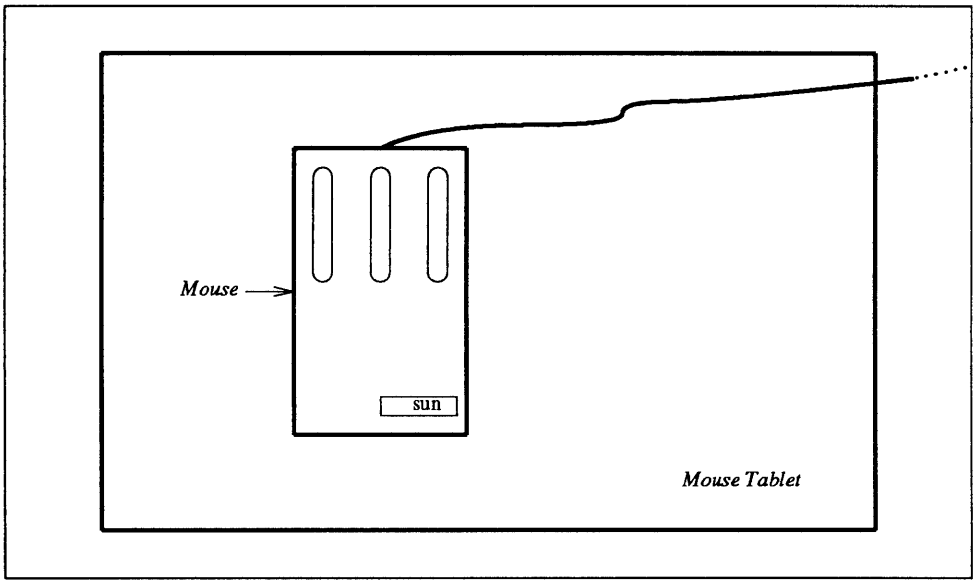
When the screen looks like this, a *window system* is running on the workstation. Each of the rectangular screen sections is a *window*.

To learn more about SunWindows, the Sun window system, read *Windows and Window-Based Tools: Beginner's Guide*.

3.3. The Mouse and the Tablet

A funny little device on a flat surface that looks like this are both parts of your workstation:

Figure 3-4 *Mouse and Tablet*



The funny little device is a *mouse* and the flat surface is its *tablet*. You can use the mouse (though it lacks fur) in conjunction with the window system on your workstation.[10]

You've completed an overview of the parts of the Sun workstation.

[10] If you are running a window system presently, you may have trouble getting what you type to appear where you want it to appear. Try to make sure that the mouse cursor (usually a long arrow) stays inside the window where what you type should appear. See *Windows and Window-Based Tools: Beginner's Guide* for more information.

4

Files and the File System

4

Files and the File System

A *file* is a "container" for your data on the system. Each file has a name. You can use files to store programs and documents for later use, much like a file cabinet. Then, you don't have to retype them each time you want to use the system. This chapter explains what the file system is, including the files and directories within it.[11]

4.1. Security: *Caveat Emptor*

UNIX provides security mechanisms for the information you store on your machine. However, except for your password, you should assume that other people may be able to read what you type and store on the machine.[12]

For more information about *file protection*, or security for files, see *Doing More With UNIX: Beginner's Guide*, especially the portions that describe file protections and the `crypt` command.

4.2. Files

You can display files, copy them, move them, edit them, and remove them.

Creating a File

Many people create files using a *text editor*, but the command `cat` allows you to create files without learning a text editor.[13]

For the purpose of learning with a standard set of files, please type the examples **as listed** here. You can experiment as you go along, but be aware that future examples often depend on these examples.

Note: Make sure to put a space character before the > sign; the space character after the > sign is optional.

Once you have logged in, you can create your first file. Type `cat >` *filename*, where *filename* is the name you wish to give the file, and RETURN.[14]

---

[11] Throughout this chapter, you may encounter error messages or other problems. When this happens, see *Self Help With Problems: Beginner's Guide*.

[12] Even once you know and use the security mechanisms UNIX offers, do not assume that a determined person could not find out vital information you store on the system. Unless you have to store confidential information on a computer, don't.

[13] A text editor is a program designed specifically to help you create and make changes to the text within a file. To learn the Sun text editor `vi`, see Chapter 6.

[14] The command `cat` is an abbreviation of concatenate, to link or connect in a series. It is one example of an admittedly obscure UNIX reference. It refers to the fact that you can look at more than one file in a row by typing more than one filename after you type `cat` (see the section on looking at a file in this chapter).

Then type the contents of the file until you are done. Even though it may seem strange, complete the file with a CTRL-D on a line by itself. For example, to create the file `alice`, with one of her famous quotations as its contents, type:

Figure 4-1 *Creating a File*

```
venus% cat > alice
"We reach for destinies beyond
what we have come to know."
^D
```

Caution: If you `cat >` into an existing file, `cat` will replace the contents of that file with the text you type.

You've created your first file!

When you try it, don't worry when CTRL-D appears as `^D` and when the command prompt (`venus%`) blots out the CTRL-D.

Note: The filename is like the label of a file folder in your desk. It's a good idea to think of a filename that expresses meaning appropriate to the the contents of the file, but is short enough so that it is easy to type.

You can name a file whatever you want, *almost.* Almost, that is, because some characters, called *special characters*, mean special things to the system. You could "confuse" the system without intending to if you used them. Special characters to watch out for are: `\`, `>`, `<`, `|`, `&`, `?`, `$`, `[`, `]`, and `*`. Don't try to put spaces in the middle of a filename. Filenames that begin with a period, like `.login`, are *hidden* files, and will not show up when you do a simple listing of files, as in the section on listing files later in this chapter.[15]

Name and create another file or two for practice.

Looking at a File

Note: CTRL-D does not appear when looking at the file. That's because the CTRL-D isn't part of the file; it's the *end-of-file* character, used to indicate to the system that the user has finished typing in a file.

Now that you have created a file successfully, you can look at it by using the `cat` command again. When you type `cat`, followed by a *filename* — this time without a `>` sign — the system will type out the file specified by that filename on the screen.

So, to look at the file `alice` created above, type:

Figure 4-2 *Looking at a File With* `cat`

```
venus% cat alice
"We reach for destinies beyond
what we have come to know."
venus%
```

Create a couple of files and look at them for practice.

[15] See *Doing More With UNIX: Beginner's Guide* for information about listing hidden files.

When you try to look at a nonexistent file with `cat`, it will type an error message. For example:

Figure 4-3 *Error: Trying to Look at a Nonexistent File*

```
venus% cat nightmare
nightmare: No such file or directory
venus%
```

You can also look at really *long* files with `cat`. One long file is `/usr/lib/units`, which describes various units used in business and science. Don't worry about the `/`'s in this filename. The section on pathnames later in this chapter will explain.

When you `cat` a really long file, it will fly by on your screen until it finishes. Only the last screenful of the file will remain.

Note: Always remember to type CTRL-Q after CTRL-S; otherwise, the system won't respond when you type.

To prevent the file from "flying by," type CTRL-S, while the system types out the file, to temporarily halt the system from typing. It will probably go by pretty fast anyway. When you want the system to start typing the rest of the file, type CTRL-Q.

So, try it:

Figure 4-4 *Looking at a Long File With* `cat`

```
venus% cat /usr/lib/units
/ dimensions
m               *a*
kg              *b*
sec             *c*
coul            *d*

... and so on until the system "hears" you type
CTRL-S , when it stops, and waits for you to type
CTRL-Q , to resume typing until the end of the file ...

tun             8 barrel
water               .22491|2.54 kg/m2-sec2
wey             40 bu
weymass             252 lb
Xunit               1.00202-13m
k               1.38047-16 erg/degC
venus%
```

You may have noticed that `cat` isn't particularly convenient for looking at long files. So, another way to look at files is with the program `more`.

Note: The amount of text on a screen will vary with the size of screen you are using.

Type `more` and the *filename*. `more` will type out only one "page," or screen of text, then it will wait. When you hit the space bar, `more` types out the next page, and so on, until you reach the end of the file.

Figure 4-5 *Looking at a File with* `more`

```
venus% more /usr/lib/units
/ dimensions
m                       *a*
kg                      *b*
sec                     *c*
coul                    *d*
candela                 *e*
dollar                  *f*
radian                  *g*
bit                     *h*
erlang                  *i*
degC                    *j*

/ constants

fuzz                    1
pi                      3.14159265358979323846
c                       2.997925+8 m/sec fuzz
g                       9.80665 m/sec2
au                      1.49597871+11 m fuzz
mole                    6.022169+23 fuzz
e                       1.6021917-19 coul fuzz
energy                  c2
force                   g
mercury                 1.33322+5 kg/m2-sec2
hg                      mercury

/ dimensionless

degree                  1|180 pi-radian
circle              2 pi-radian
turn            2 pi-radian
grade               .9 degree
--More--(5%)
```

Type the space bar to see more of the file or type `q`*, for quit, to stop looking at it.*

Note: `more` tells you the percentage (rounded to nearest integer) of the file text it has typed onto the screen.

`more` doesn't let the file "fly by" like `cat` does, so `more` is especially good for looking at long files.

Using `more`, look at the files you have created.

Printing a File

You've created and looked at a file, now you can print one. To print the file `alice`, type `lpr` and the filename, `alice`:

Figure 4-6 *How to Print Out a File*

```
venus% lpr alice
venus%
```

Simple right? Sometimes, it's not quite so simple. You may have to:

- Find out where the printer is, assuming there is a printer.
- Find out which of a few printers your file is printing on.
- Add paper to, or fix a paper jam in, the printer.
- Get someone to tackle a printer software problem.

When something goes wrong, ask someone for help; when you can't, see *System Administration for the Sun Workstation.*

The result is a *printout*, or *hardcopy*, of your file.

Listing Files

To get a list of the files you have created, `alice` and a the file `emma` that you create in this example, type `ls` (for "list"):[16]

Figure 4-7 *Listing Files*

```
venus% cat > emma
"If I can't dance, I don't want to
be part of your revolution."
 (Don't forget to end with CTRL-D.)
venus% ls
alice   emma
venus%
```

`ls` displays the filenames in alphabetical order.

Moving (or Renaming) Files

To *rename* one of your files, type `mv` (for "move") followed by the current *filename* and the new *filename*. For example, to rename the file `alice` as `walker`, type:

[16] See *Doing More With UNIX: Beginner's Guide* for information about listing *hidden* files, files that have names beginning with the period character.

Figure 4-8 *Moving and Renaming Files*

```
venus% ls
alice   emma
venus% mv alice walker
venus% ls
emma    walker
venus%
```

In the example, `ls` returns a file listing, indicating that the file was renamed successfully.

When you try to `mv` a file that doesn't exist, `mv` types an error message and doesn't move anything:

Figure 4-9 *Error: Trying to Move a File That Doesn't Exist*

```
venus% ls
emma    walker
venus% mv mirage water
mv: cannot access mirage
venus% ls
emma    walker
venus%
```

When you try to `mv` a file onto an existing file, `mv` overwrites the contents of the target file with the contents of the file you are moving. For example, when you move the file `walker` onto the file `emma`, `mv` removes the contents of `emma` and replaces them with the contents of `walker`. In other words, the contents of the file `emma` disappear, replaced by the contents of the file `walker`.

So, **be careful** when moving files onto other files.[17]

[17] In the special case when you try to move a file onto itself, `mv` types an error message and doesn't move anything:

```
venus% mv walker walker
mv: walker and walker are identical
venus%
```

Figure 4-10 **Caution**: *Moving a File to an Existing Filename*

```
venus% ls
emma    walker
venus% cat emma
"If I can't dance, I don't want to
be part of your revolution."
venus% mv walker emma
venus% ls
emma
venus% cat emma
"We reach for destinies beyond
what we have come to know."
venus%
```

Copying Files

To copy one of your files, type `cp` (for "copy") followed by the *filename* of the file you want copied and the *filename* of the file that is the copy. For example, to make a copy of the file `emma`, that copy called `hurston`:

Figure 4-11 *Copying a File*

```
venus% ls
emma
venus% cp emma hurston
venus% ls
emma    hurston
venus% cat emma
"We reach for destinies beyond
what we have come to know."
venus% cat hurston
"We reach for destinies beyond
what we have come to know."
venus%
```

`cp` copies *from* the first file *to* the second. In this example, `cp` copies the contents of file `emma` into the new file `hurston`. *Unlike* `mv`, `cp` keeps the first file intact, making a copy of it without altering it.

Just as with mv, when you try to cp a file that doesn't exist (void in the next example), cp types an error message and doesn't copy anything:

Figure 4-12 *Error: Trying to Copy a File That Doesn't Exist*

```
venus% ls
emma    hurston
venus% cp void existence
cp: void: No such file or directory
venus% ls
emma    hurston
venus%
```

When you try to cp from one file to an existing file, cp replaces the contents of the target file with the contents of the file you are copying. For example, when you create a file called wilde, and copy it onto the file hurston, cp removes the contents of hurston and replaces them with the contents of wilde. *Unlike* mv, however, the file wilde doesn't disappear when cp copies its contents into the file hurston.[18]

Figure 4-13 **Caution**: *Copying a File to an Existing Filename*

```
venus% cat > wilde
"There is no such thing as a moral or an immoral
book.  Books are well written or badly written.
That is all."
 (Don't forget to end with CTRL-D.)
venus% ls
emma    hurston wilde
venus% cat hurston
"We reach for destinies beyond
what we have come to know."
venus% cp wilde hurston
venus% ls
emma    hurston wilde
venus% cat hurston
"There is no such thing as a moral or an immoral
book.  Books are well written or badly written.
That is all."
venus%
```

[18] In the special case when you try to copy a file onto itself, cp types an error message and doesn't copy anything:

```
venus% cp hurston hurston
cp: Cannot copy file to itself.
venus%
```

Removing (or Deleting) Files

With all of these files around, it gets really messy. The `rm` command allows you to clean up by removing, or deleting, the files you no longer need.

Take care in using the `rm` command. Once you `rm` something, you may never get it back.[19] If you wish to make `rm` a bit more cautious about removing your files, see *Setting Up Your UNIX Environment: Beginner's Guide.*
To remove a file, type `rm` followed by the *filename(s)* of the file(s) you wish to remove. Since, after trying `mv` and `cp`, the files `emma` and `hurston` both contain quotations not said by them, to `rm` them type:

Figure 4-14 *Removing (or Deleting) a File*

```
venus% ls
emma    hurston wilde
venus% rm emma hurston
venus% ls
wilde
venus%
```

Notice that you can remove more than one file at once by including filenames of all the files you wish to remove after the `rm` command. Many of the UNIX commands accept more than one filename.

When you try to `rm` a file that doesn't exist, `rm` types an error message and doesn't remove anything:

Figure 4-15 *Error: Trying to Remove a Nonexistent File*

```
venus% ls
wilde
venus% rm gryphon
rm: gryphon nonexistent
venus%
```

4.3. Directories

Just as a file is a container for text or programs, a *directory* is a container for files. You could think of a directory as a filing cabinet that can store files for you. You can create a directory to store all of the files for a certain project.

Directories act as glue to link the UNIX file system together. Create, delete, and use them as you need them.

[19] If you or your system administrator does *backups* of your file system, you may be able to retrieve deleted files. Contact your system administrator or look at *System Administration for the Sun Workstation.*

Creating (or Making) Directories

To make a directory, type `mkdir` followed by the new *directory name.* For example, to create the directory `quotations`:

Figure 4-16 *Creating (or Making) a Directory*

```
venus% mkdir quotations
venus% ls
quotations      wilde
venus%
```

When you list your files, you can see that the directories appear in the listing along with the files. In fact, directories are a kind of file — a file that can store other files, including other directories.[20]

When a file and a directory appear in a listing together, it does **not** mean that the directory contains the file; see the section on listing files in other directories in this chapter.

When you try to `mkdir` a directory that already exists, `mkdir` types an error message and doesn't make any directories:

Figure 4-17 *Error: Trying to Create an Existing Directory*

```
venus% ls
quotations      wilde
venus% mkdir quotations
mkdir: quotations: File exists
venus%
```

Removing Empty Directories

To remove empty directories, in other words, directories that don't contain files or other directories, use the `rmdir` command. For example, to remove the directory `quotations`:

Figure 4-18 *Removing an Empty Directory*

```
venus% ls
quotations      wilde
venus% rmdir quotations
venus% ls
wilde
venus%
```

You can remove more than one directory at a time by typing `rmdir` followed by the directory names of all of the directories you want to remove.

[20] If you want to be able to distinguish easily your directories from your files when you list them, see the section on using an option to `ls` in Chapter 10.

Removing Non-Empty Directories

When you try to remove a directory that contains some files or directories, `rmdir` types an error message and doesn't remove anything. For this example, assume that directory `bloated` contains files or directories.

Figure 4-19 *Error: Trying to Remove a Directory and Its Contents with* `rmdir`

```
venus% ls
bloated wilde
venus% rmdir bloated
rmdir: bloated: Directory not empty
venus%
```

Removing Non-Existent Directories

When you try to remove a directory that doesn't exist, `rmdir` types an error message and doesn't remove anything:

Figure 4-20 *Error: Trying to Remove a Nonexistent Directory*

```
venus% ls
wilde
venus% rmdir black.hole
rmdir: black.hole: No such file or directory
venus%
```

Removing Something That Isn't a Directory

When you try to `rmdir` a file that isn't a directory, `rmdir` types an error message and doesn't remove anything:

Figure 4-21 *Error: Trying to Remove a File With* `rmdir`

```
venus% ls
wilde
venus% rmdir wilde
rmdir: wilde: Not a directory
venus%
```

To remove a directory that contains files or directories, see the section on removing directories and their contents later in this chapter.

Moving and Copying Files to New Directories

Now that you know how to make directories, you may want to store some files in them. You can `mv` and `cp` files into your directories.

Note: Notice the *underscore character* in the directory name `famous_people`.

Make a directory `famous_people` and a file `actresses`, and move the new file into the new directory by typing `mv` followed by the *filename* and the *directory name*:

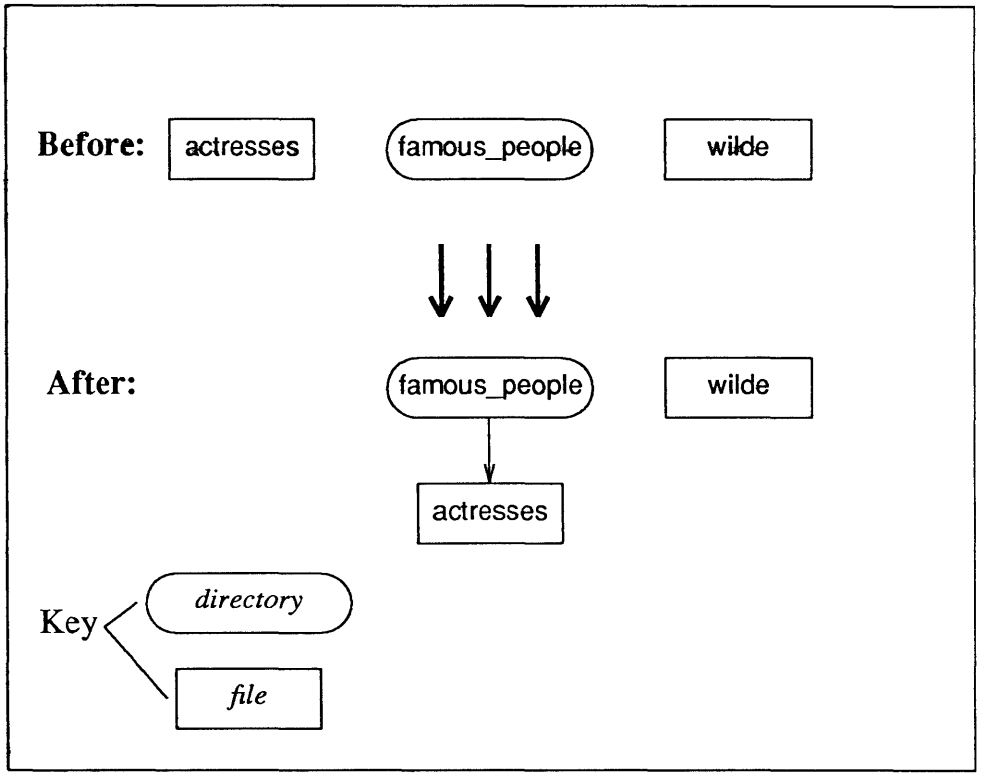
Figure 4-22 *Moving a File into a Directory*

```
venus% ls
wilde
venus% mkdir famous_people
venus% cat > actresses
Marilyn Monroe
Katharine Hepburn
 (Don't forget to end with CTRL-D.)
venus% ls
actresses       famous_people   wilde
venus% mv actresses famous_people
venus% ls
famous_people   wilde
venus%
```

Notice that the file you moved into the new directory does not show up when you list your files. To see it, you will have to list the files in that directory (see the section on listing files in other directories later in this chapter).

But, for now, be confident that you have moved the file `actresses` into the directory `famous_people`, as shown in this diagram:

Figure 4-23 *Effects of Moving a File into a Directory*



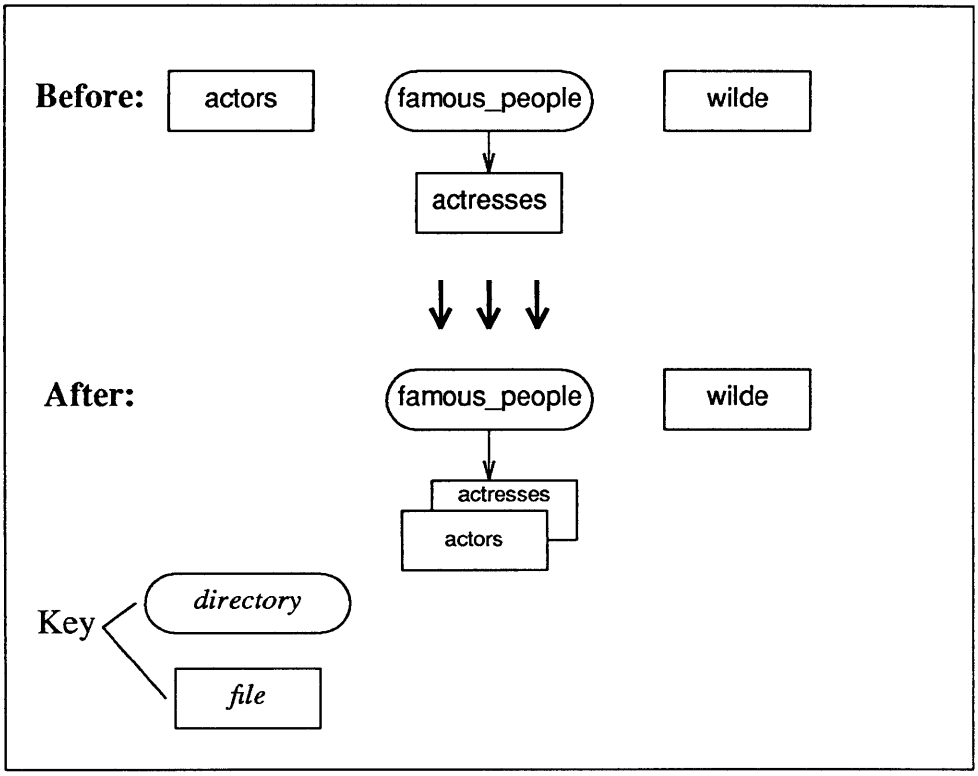
Next, let's create a new file `actors` and copy it into the directory `famous_people`:

Figure 4-24 *Copying a File into a Directory*

```
venus% ls
famous_people   wilde
venus% cat > actors
James Dean
Humphrey Bogart
 (Don't forget to end with CTRL-D.)
venus% cp actors famous_people
venus% ls
actors          famous_people   wilde
venus%
```

The file `actors` appears in your list of files, but a copy of it is in the `famous_people` directory too.

Figure 4-25 *Effects of Copying a File into a Directory*



`mv` and `cp` will give the same error messages as before when you try to move and copy from nonexistent files, or onto existing files.

Listing Files in Other Directories

To list the files in other directories, type `ls` followed by the *directory name* of the directory you want to list:

Figure 4-26 *Listing the Files in Another Directory*

```
venus% ls
actors          famous_people   wilde
venus% ls famous_people
actors          actresses
venus%
```

The file `actors` appears in the both listings because you copied it into the directory `famous_people`.

When you list the files in a directory that has no files, `ls` doesn't list any files. Notice that no two files in the same directory can have the same filename.

When you try to list the files in a nonexistent directory, `ls` types out an error message and doesn't list anything:

Figure 4-27 *Error: Trying to List a Nonexistent Directory*

```
venus% ls
actors          famous_people   wilde
venus% ls plot
plot not found
venus%
```

When you list a file, rather than a directory, `ls` types out the filename of that file:

Figure 4-28 *Listing a File, Not a Directory*

```
venus% ls
actors          famous_people   wilde
venus% ls wilde
wilde
venus%
```

Moving (or Renaming) Directories

Since it is possible to move and copy files, why not move and copy directories?

To move (or rename) a directory, type `mv` followed by the old *directory name* and the new *directory name*. For example, to move the directory `famous_people` into the directory `film.people`:

Figure 4-29 *Moving (or Renaming) a Directory*

```
venus% ls
actors          famous_people   wilde
venus% mv famous_people film.people
venus% ls
actors          film.people     wilde
venus% ls film.people
actors          actresses
venus%
```

You may have wondered why the command for renaming files and directories is `mv`, rather than perhaps `rn`, for "rename." The reason is that you can actually move one directory into another directory. The directory that you move into the other directory is a *subdirectory* of that directory. Extending the analogy of directory as filing cabinet, you could think of a subdirectory as one drawer of the filing cabinet.

Make a directory and move or copy files into it. Then, move the directory into another directory by typing `mv` followed by the *directory name* of the directory you want to move (with its contents) and the *directory name* of the directory into which you want to move it. For example:

Figure 4-30 *Moving a Directory into Another Directory*

```
venus% mkdir directors
venus% cat > spielberg
Jaws
Close Encounters of the Third Kind
E.T.
Back to the Future
The Color Purple  (Don't forget to end with CTRL-D.)
venus% mv spielberg directors
venus% ls
actors          directors       film.people     wilde
venus% ls directors
spielberg
venus% ls film.people
actors          actresses
venus% mv directors film.people
venus% ls
actors          film.people     wilde
venus% ls film.people
actors          actresses       directors
venus%
```

Remember that the file `spielberg` is not in the directory `directors` when you create it. For now, a file won't appear in a directory you've created until you explicitly move it there.

When you try to move a directory into a file, `mv` will type an error message and won't move anything:

Figure 4-31 *Error: Trying to Move a Directory into a File*

```
venus% mv film.people wilde
mv: film.people: rename: Not a directory
venus%
```

The system checks to make sure you don't move a directory into a file, so the system won't remove the file when you try to move a directory onto it.

Removing Directories and Their Contents

When you want to remove a directory and its contents, the easiest way is to type `rm -r` and the *directory name* of the directory you want to remove. For example, to remove the directory `film.people` and its contents:

Figure 4-32 *Removing a Directory and Its Contents*

```
venus% ls
actors          film.people     wilde
venus% rm -r film.people
venus% ls
actors  wilde
venus%
```

The `-r` in `rm -r` is an *option* to the `rm` command. Learn more about options in the section on options in Chapter 5 . The `-r` option to `rm` causes `rm` to execute recursively, that is, `rm` removes the file or directory you specify, the *contents* of the directory you specify, the contents of that directory, and so on.

Be especially careful about using `rm -r`, especially in combination with the special character `*`, because you may delete files and directories that you intend to save.

Changing Working Directories

When you list files, move or copy files into another directory, and list the files in that directory, you may have wondered: Where did I start out? What am I moving and copying these files from?

The answer is: When you first log in, you are in a special directory called your *home directory*. In fact, every time you log in you will start out in your home directory.

So far, you have done all of your work while in your home directory. But you can change directories to do work in other directories too.

Current Working Directory

The directory in which you are working is your current directory, or your *working directory*. To change your working directory from your home directory to one of the directories you have created, type `cd` and the *directory name* of the directory you want to become your working directory. For example, to change directories from your home directory (where you are now) to a newly-created directory `hinterland`:

Figure 4-33 *Changing Your Working Directory*

```
venus% mkdir hinterland
venus% ls
actors          hinterland      wilde
venus% ls hinterland
venus% cd hinterland
venus% ls
venus%
```

As you can see, when you list the files in your working directory, you get a list of the files (no files) in the directory you changed to.

You can type any of the other commands you have learned in this new working directory, just like when you're in your home directory. Try it!

Getting to Your Home Directory

You can get back to your home directory at any time by typing `cd`, without a *directory name*. So, don't worry when you get lost in a maze of files and directories, you can always return home with `cd`. For example, when you are in the empty directory `hinterland`, type `cd` and you will return to your home directory:

Figure 4-34 *Changing Directories Back to Your Home Directory*

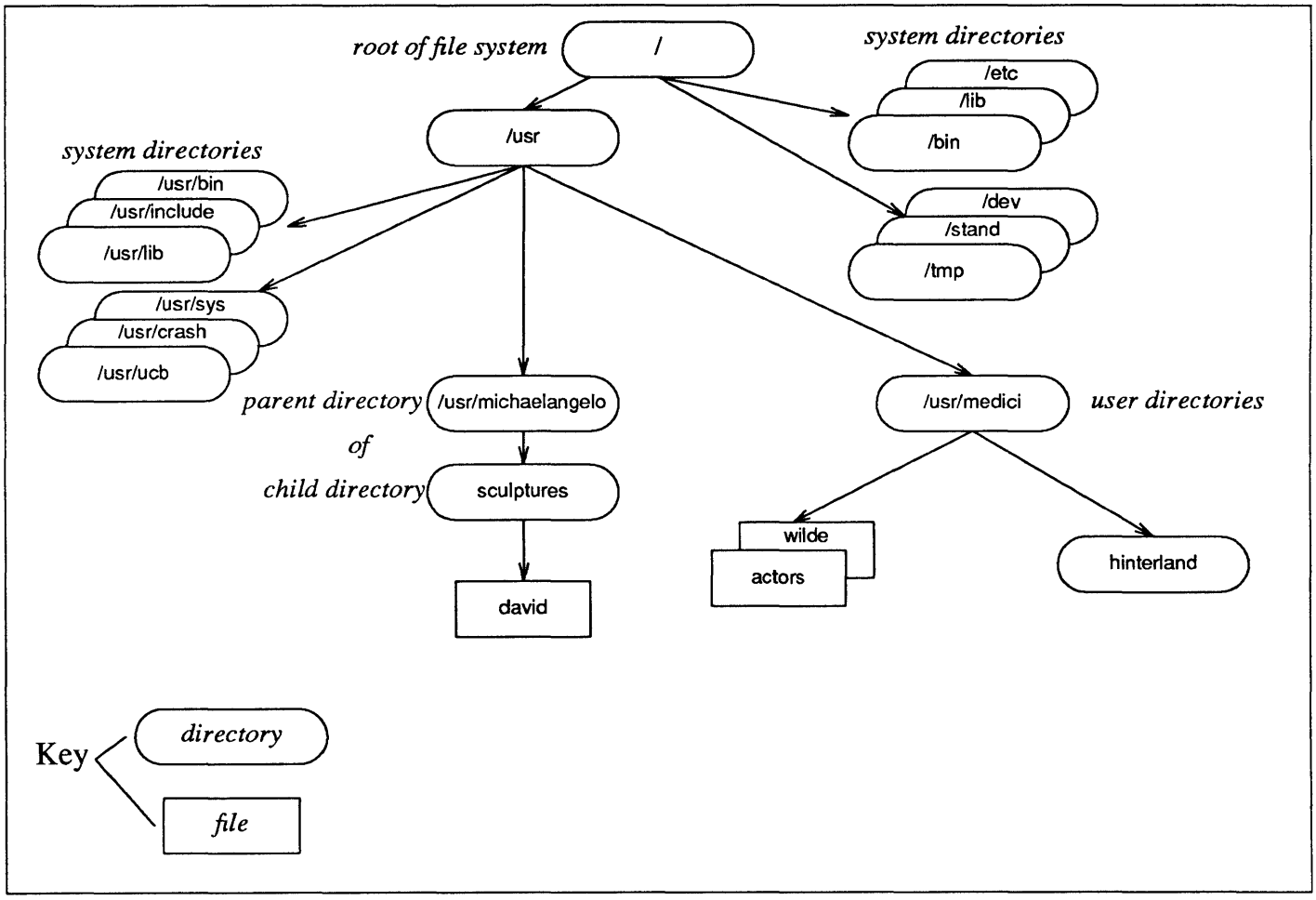
```
venus% ls
venus% cd
venus% ls
actors          hinterland      wilde
venus%
```

You can also travel into other user's directories, directories accessible by groups of users, and directories used to maintain the system. If you travel outside of your own directories, **do not modify** the files and directories you find there, unless you are absolutely sure you should do so.

4.4. The File System Hierarchy

All of the files and directories on the system form the *file system hierarchy*. This hierarchy has a *tree structure*. Somewhat paradoxically, at the "top" of the tree is the *root directory*. The root directory is the *parent directory* of its subdirectories, or *child directories*. Each of the subdirectories acts as a parent directory to its own child directories (if it has any directories "below" it) throughout the tree structure.

Figure 4-35 *The UNIX File System Tree Structure*



The big advantage of the UNIX file system is that you can customize the structure of files and directories to meet your own needs.

To find out what the actual Sun file system hierarchy on your system looks like, see the description of the `hier` Man Page in *Setting Up Your UNIX Environment: Beginner's Guide.*

What Directory Is This?

At times, it is useful to know where you are in the file system hierarchy — that is, what directory is your working directory.

To find out your working directory, type `pwd` (for "print working directory"):

Figure 4-36 *How to Find Out Your Working Directory Name*

```
venus% pwd
/usr/medici
venus% cd hinterland
venus% pwd
/usr/medici/hinterland
venus% cd
venus% pwd
/usr/medici
venus%
```

Notice how `cd`, without a *directory name*, returns you to your home directory.

Don't worry if your working directory doesn't look like the working directory in this example. You have to understand *pathnames*, explained in the next section, to make sense of it all.

Pathnames: Relative and Absolute

Some logical questions to ask at this point are:

- Why does `pwd` give me such a strange working directory name?
- What is this bizarre thing with all the slashes?

Note: Although home directory pathnames are fairly standard, you or your system administrator may modify the file system so that your home directory appears in a different portion of the hierarchy.

Since your home directory is part of the file system hierarchy, it occupies a certain place in that hierarchy. UNIX uses *pathnames*, or branches, with a certain notation, to indicate a location in the hierarchy tree.

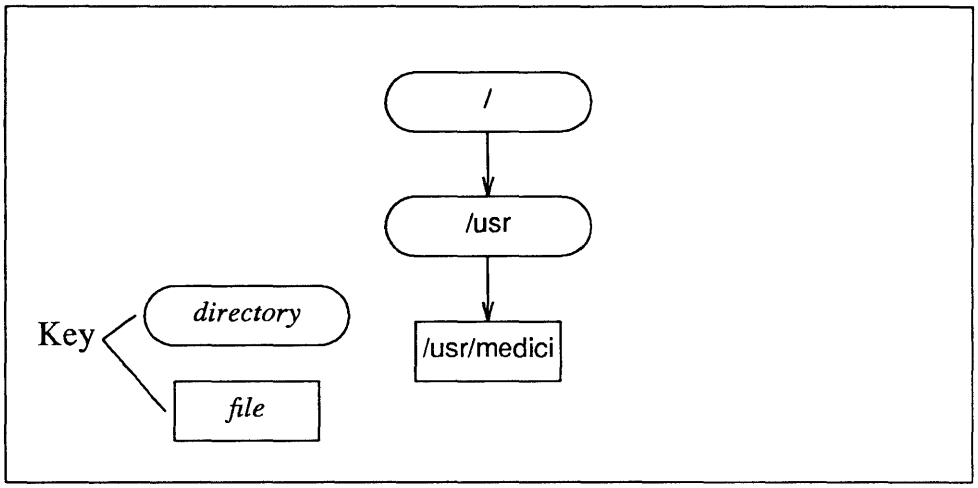
There are three kinds of pathnames: *simple*, *absolute*, and *relative*. When you learn to choose the right type of pathname for the task, your work becomes easier.

So far, you have typed only simple pathnames. Simple pathnames are file or directory names that don't include any complex information about position of the file or directory within the file system hierarchy. So, when you type `cd hinterland`, `hinterland` is a simple pathname. `cd` takes `hinterland` to mean the directory `hinterland` that is a child of the working directory.

Absolute pathnames indicate the absolute position of a file or directory within the hierarchy. They begin with the slash character, `/`, denoting the root directory of the entire file system. The absolute pathname tells you "how to get there from the system root directory."

The root directory has a child directory called `usr`. The traditional home directory pathname of user `medici` on workstation `venus` is `/usr/medici`.

Figure 4-37 *Absolute Pathname of Home Directory*



Absolute Pathnames

The absolute pathname traces the absolute position of a file or directory in the file system hierarchy. It starts with a / to represent the root directory, followed by its child directory and another / to separate the name of the child directory from the name of its own child, and so on. Don't be confused by the two meanings of / : it's both the name of the root directory and a separator in pathnames.

Another way to change directories to your home directory is to type `cd` followed by the absolute pathname of your home directory (although typing `cd` alone is more efficient):

Figure 4-38 *Using the Absolute Pathname of Your Home Directory*

```
venus% pwd
/usr/medici
venus% cd hinterland
venus% pwd
/usr/medici/hinterland
venus% cd /usr/medici
venus% pwd
/usr/medici
venus%
```

Relative Pathnames

Relative pathnames differ from absolute pathnames in that they don't provide an absolute path within the file system hierarchy. They don't trace the file or directory position from the root directory down through its children and their children until the desired file or directory.

Instead, relative pathnames trace the path from the working directory to the desired file or directory, or "how to get there from here." They are most useful when trying to get from one file or directory on a branch to another file or

directory on that branch.

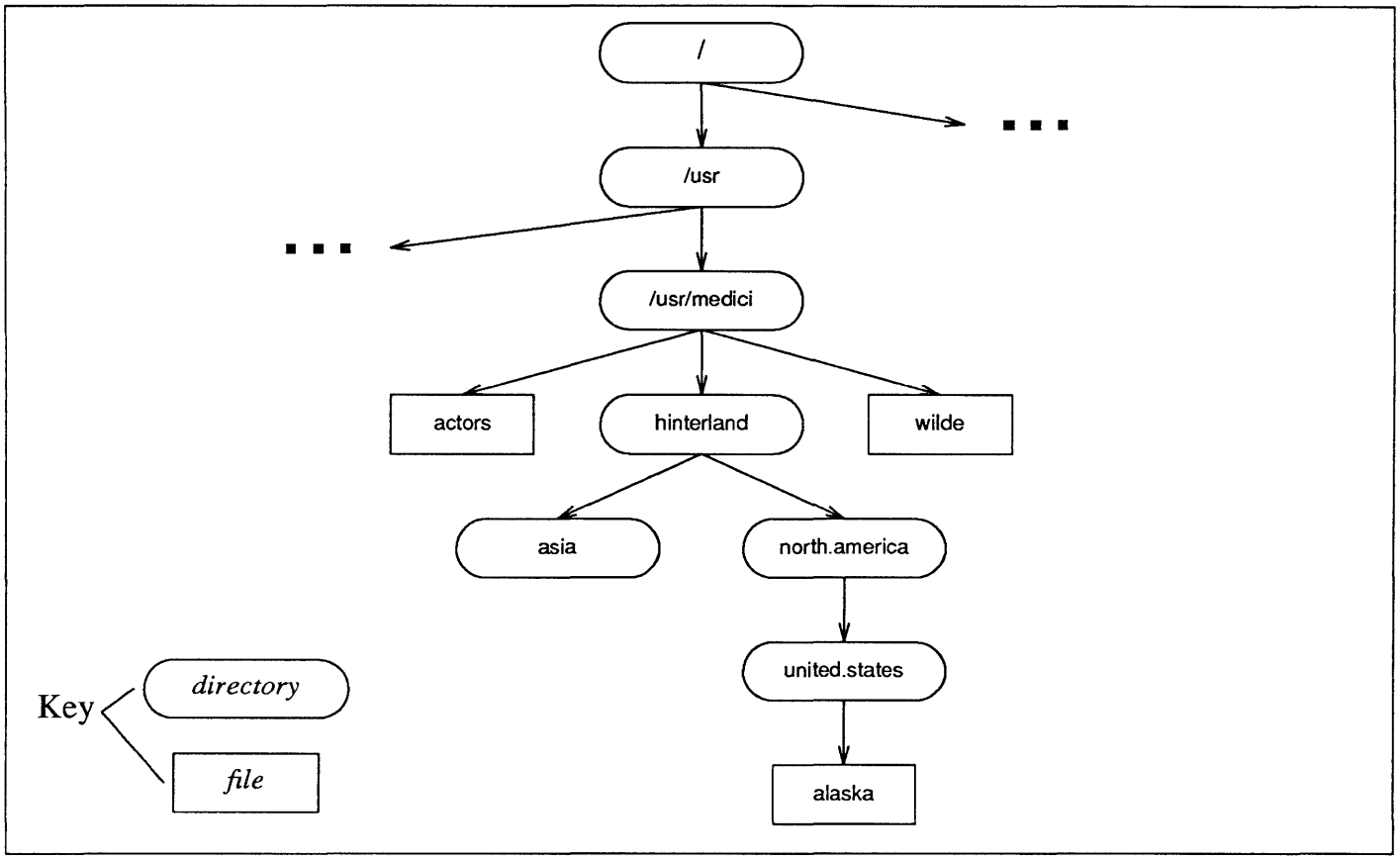
To experiment with relative pathnames, construct your own little tree of files and directories. Do something like this:

Figure 4-39 *Setting Up a Tree of Files and Directories*

```
venus% pwd
/usr/medici
venus% cd hinterland
venus% mkdir asia
venus% mkdir north.america
venus% cd north.america
venus% mkdir united.states
venus% cd united.states
venus% cat > alaska
Alaska is the largest state in the United States.
 (Don't forget to end with CTRL-D.)
venus% cd
venus% pwd
/usr/medici
venus%
```

Here's what the file and directory tree looks like:

Figure 4-40 *A File and Directory Tree*



To access a file or directory from another directory, it is not necessary to be in that directory or its parent, or to specify the absolute pathname. In fact, if one had to use only simple or absolute pathnames all the time, some situations would require a lot of typing and inconvenience.

When accessing a distant file or directory, the relative pathname permits you to specify only those parts directly on the path between your current location and the location of the object you want to access.

So, in our example tree structure, to `cat` the file `alaska` from the working directory `hinterland`, type `cat` followed by the relative pathname of the file in relation to the working directory:

Figure 4-41 *Using Relative Pathnames to Look at a File in Another Directory*

```
venus% pwd
/usr/medici
venus% cd hinterland
venus% pwd
/usr/medici/hinterland
venus% ls /usr/medici/hinterland/north.america/united.states
alaska
venus% cat north.america/united.states/alaska
Alaska is the largest state in the United States.
venus% pwd
/usr/medici/hinterland
venus%
```

In the above example, `ls` uses the absolute pathname of the directory, and `cat` uses the relative pathname of the desired file in that directory. Clearly, the relative pathname is more convenient in this instance.

Abbreviations for Special Directory Pathnames

Because they get typed so often, special directories, such as the home directory, current directory, and parent directory, have abbreviated notations for their pathnames.

Home Directory Abbreviation

You can specify your home directory pathname as the *tilde* character, or "squiggle": ~ . When you are busy in a working directory somewhere distant in the tree structure and you want to list the files in your home directory, you can use the ~ abbreviation. Substitute ~ for the specification of your home directory pathname:

Figure 4-42 *Abbreviation of the Home Directory in a Pathname*

```
venus% pwd
/usr/medici/hinterland
venus% cd north.america/united.states
venus% pwd
/usr/medici/hinterland/north.america/united.states
venus% ls ~ (Notice the squiggle here.)
actors          hinterland      wilde
venus% cd
venus% pwd
/usr/medici
venus% ls
actors          hinterland      wilde
venus%
```

The advantage of using the abbreviation is that you type fewer characters and you can remain in your working directory where you may have easier access to the files and directories you're working on.

You can access other user's home directories by typing the tilde character, ~ , followed by that person's username. So to connect to user pandora's home directory, type cd followed by ~*username*, or in this case:[21]

Figure 4-43 *Abbreviation of Other User's Home Directories in a Pathname*

```
venus% pwd
/usr/medici
venus% cd ~pandora
venus% pwd
/usr/pandora
venus% cd
venus% pwd
/usr/medici
venus%
```

The next example is difficult, but persist and you will understand.

Working Directory Abbreviation

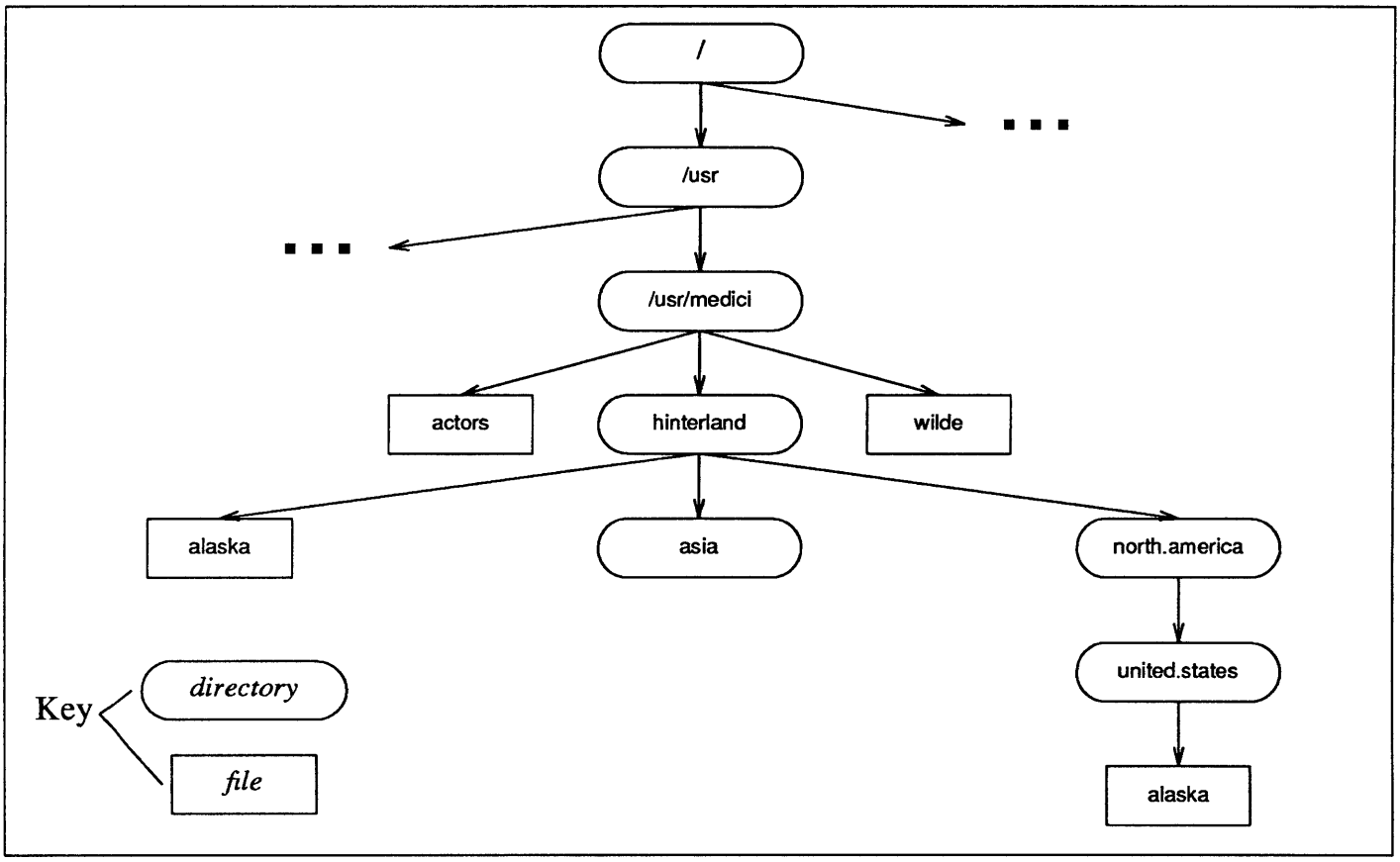
You can specify the working directory pathname as the dot (.). This abbreviation is useful in a variety of situations. For instance, when you want to copy a file from a distant directory in the file system hierarchy into your working directory, you can type cp followed by the pathname of the file and the working directory abbreviation (.):

Figure 4-44 *Abbreviation of the Working Directory in a Pathname*

```
venus% pwd
/usr/medici
venus% cd hinterland
venus% pwd
/usr/medici/hinterland
venus% ls
asia            north.america
venus% ls north.america/united.states
alaska (Notice the dot at the end of the next line.)
venus% cp north.america/united.states/alaska .
(In this case, . means /usr/medici/hinterland.)
venus% ls
alaska          asia            north.america
venus% pwd
/usr/medici/hinterland
venus%
```

Here's what the tree looks like after copying file `alaska`:

Figure 4-45 *File and Directory Tree Modified by File Copying*



Parent Directory Abbreviation

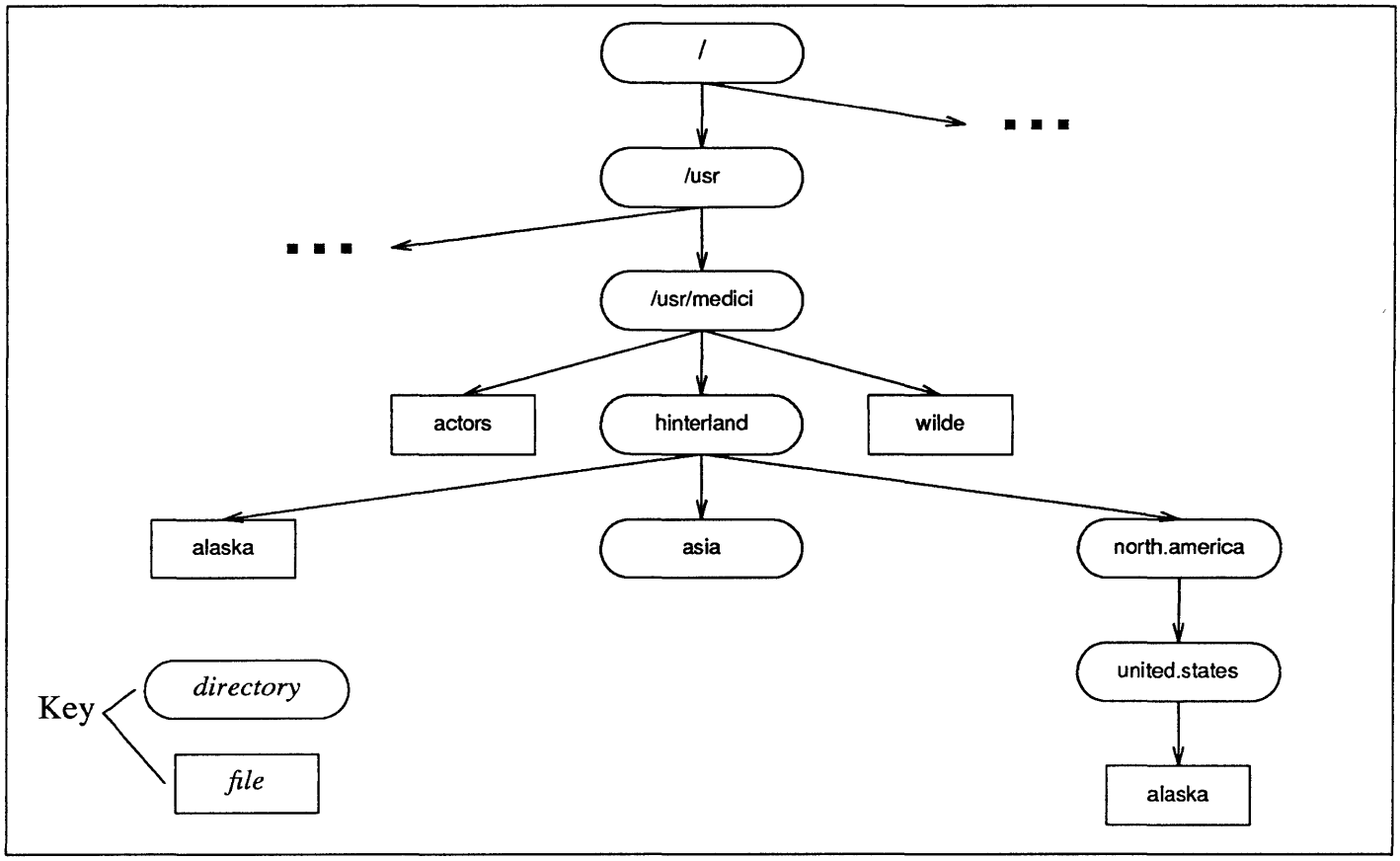
You can specify the pathname of the parent directory of the working directory as two consecutive dots (. .). As with the other abbreviations, this abbreviation makes it easier to get certain things done. For example, to move a file from your working directory into its parent directory, type `mv` followed by the filename of the file you want to move and the abbreviation for the file's destination, the parent directory (. .):

Figure 4-46 *Abbreviation of the Parent Directory in a Pathname*

```
venus% pwd
/usr/medici/hinterland
venus% ls ~ (Notice the squiggle here.)
actors          hinterland       wilde
venus% ls
alaska          asia             north.america
venus% mv alaska .. (Notice the two dots here.)
venus% ls
asia            north.america
venus% ls .. (Notice the two dots here.)
actors          alaska           hinterland      wilde
venus% cd
venus% pwd
/usr/medici
venus%
```

Here's what the tree looks like after moving file `alaska`:

Figure 4-47 *File and Directory Tree Modified by File Moving*



Parent Directory Abbreviation

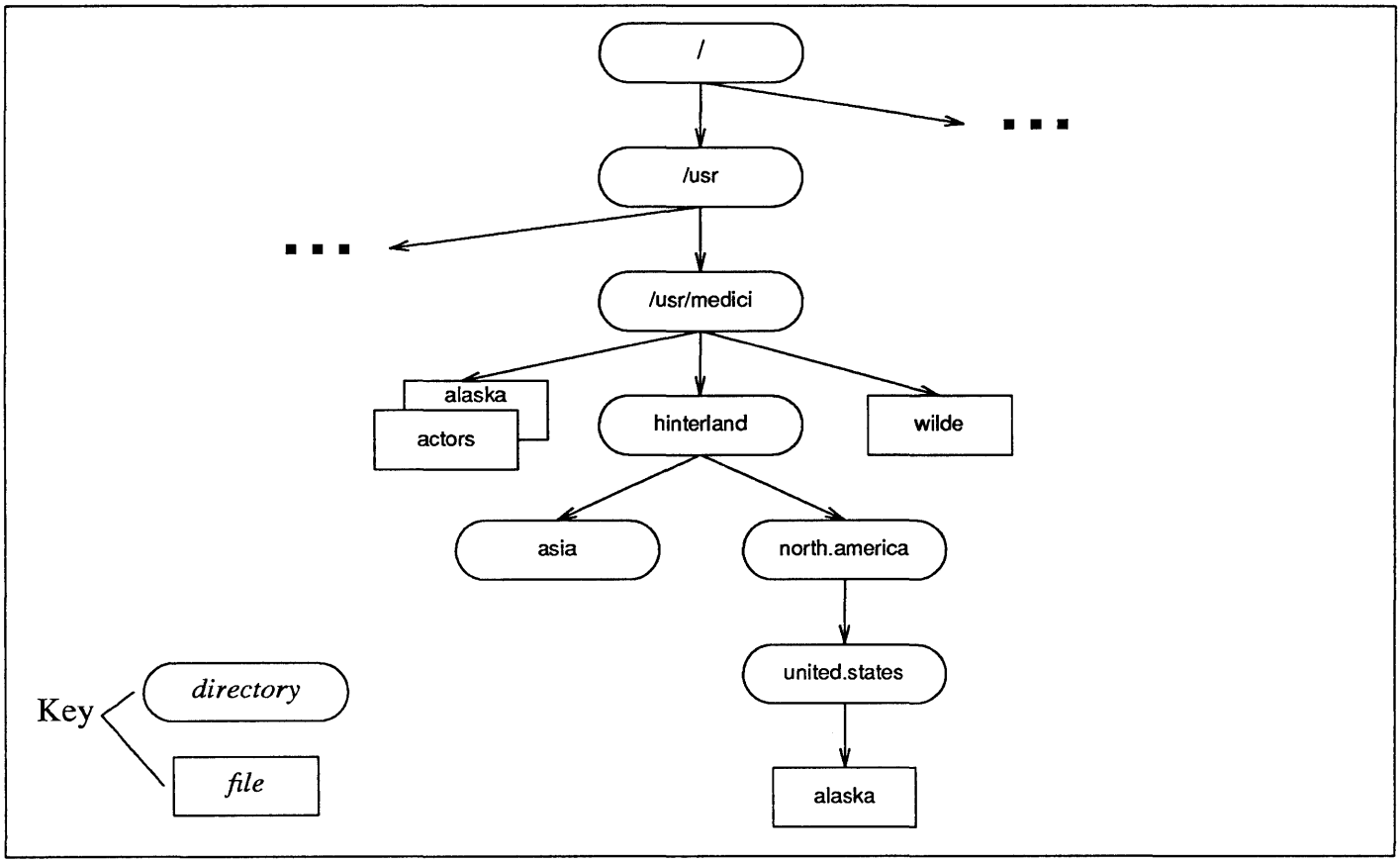
You can specify the pathname of the parent directory of the working directory as two consecutive dots (`..`). As with the other abbreviations, this abbreviation makes it easier to get certain things done. For example, to move a file from your working directory into its parent directory, type `mv` followed by the filename of the file you want to move and the abbreviation for the file's destination, the parent directory (`..`):

Figure 4-48 *Abbreviation of the Parent Directory in a Pathname*

```
venus% pwd
/usr/medici/hinterland
venus% ls ~ (Notice the squiggle here.)
actors          hinterland       wilde
venus% ls
alaska          asia             north.america
venus% mv alaska .. (Notice the two dots here.)
venus% ls
asia            north.america
venus% ls .. (Notice the two dots here.)
actors          alaska           hinterland      wilde
venus% cd
venus% pwd
/usr/medici
venus%
```

Here's what the tree looks like after moving file `alaska`:

Figure 4-49 *File and Directory Tree Modified by File Moving*



Notice that two abbreviations may serve the same purpose, as with the listing of the parent directory, that happened to be the home directory in the example above.

Perhaps the most common use of the parent directory abbreviation is to change directories into the parent directory of your working directory.

Figure 4-50 *Changing Working Directory Into Parent Directory*

```
venus% pwd
/usr/medici
venus% cd hinterland/north.america
venus% pwd
/usr/medici/hinterland/north.america
venus% cd ..(Notice the two dots.)
venus% pwd
/usr/medici/hinterland
venus%
```

4.5. Brief Review

Here is a table that provides a brief review of the last two sections (assume that your working directory is `/usr/medici/hinterland` when pertinent):

Table 4-1 *Directory Types and Abbreviations*

Syntax	*Definition*	*Example*
directory_name	simple directory name	`north.america`
root_directory/child/child/...	absolute directory name	`/usr/medici/hinterland/north.america/united.states`
child_directory/child/...	relative directory name	`north.america/united.states`
~	home directory	`usr/medici`
.	working directory	`/usr/medici/hinterland`
..	parent directory	`/usr/medici`

You've learned a good deal about the file system. If you want it, you deserve a break at this point, so you're refreshed to start out the discussion of commands in the next chapter.

5

Commands

5

Commands

You type *commands* to tell the system what to do. In past chapters, you learned commands like `cat`, `cp`, and `mv`. In this chapter, you will learn the structure of commands, so that you can understand many more of them without learning all of the structural details each time.

5.1. Command Grammar

Since you've been using some commands, you may have noticed that commands have a certain grammar. One could draw an analogy between UNIX command syntax and English, or other *natural languages*.[22]

Note: Technically, the *command* is part of a *command line* that includes the command and its options, filenames, and other expressions.

All parts of UNIX *command lines* correspond to English parts of speech. A command line consists of the actual command and its *arguments*, the rest of the command line. The command is similar to a verb. Of the arguments, *options* are similar to adverbs, and *filenames* or other *expressions* are similar to objects of the verb.[23]

Two Simple Commands

`date` is a useful command that types the current date and time associated with your geographical location.[24]

If you decided to ask for the current date and time at the stroke before midnight on the last New Year's Eve of the century, you would type `date`:

Figure 5-1 *The* `date` *Command*

```
venus% date
Fri Dec 31 23:59:59 PST 1999
venus%
```

[22] Natural languages are languages that people commonly speak, read, or write within a society or culture.

[23] Interestingly, the noun as subject of the UNIX "sentence" doesn't exist. One may assume that commands are interrogative, so it isn't necessary to preface commands with "you," meaning the system.

For further information on the analogy between UNIX and natural language, see *Thoughts on an All-Natural User Interface*, by Marion O. Harris, pp. 343-347 in the USENIX Conference proceedings for Summer 1985.

[24] Don't ask what happens in outer space! If your date and time don't reflect the appropriate date and time for your location, contact your system administrator or see *System Administration for the Sun Workstation*.

`PST` is an abbreviation for Pacific Standard Time, from the time zone in which this manual was written. To get the Greenwich Mean Time, or u*niversal time*, type the command `date`, followed by the option `-u`:

Figure 5-2 *The* `date` *Command with Option for Universal Time (GMT)*

```
venus% date -u
Sat Jan 1  7:59:59 GMT 2000
venus%
```

Another useful command, `cal`, has one required argument, one optional argument, and no command options. The required argument is the *year* for which you want to construct a calendar. The optional argument is the *month*. So, you can construct a calendar for the whole year of 2001 by typing:

Figure 5-3 *Constructing a Yearly Calendar*

```
venus% cal 2001


                                2001

          Jan                   Feb                   Mar
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
    1  2  3  4  5  6                 1  2  3                 1  2  3
 7  8  9 10 11 12 13    4  5  6  7  8  9 10    4  5  6  7  8  9 10
14 15 16 17 18 19 20   11 12 13 14 15 16 17   11 12 13 14 15 16 17
21 22 23 24 25 26 27   18 19 20 21 22 23 24   18 19 20 21 22 23 24
28 29 30 31            25 26 27 28            25 26 27 28 29 30 31

          Apr                   May                   Jun
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
 1  2  3  4  5  6  7          1  2  3  4  5                 1  2
 8  9 10 11 12 13 14    6  7  8  9 10 11 12    3  4  5  6  7  8  9
15 16 17 18 19 20 21   13 14 15 16 17 18 19   10 11 12 13 14 15 16
22 23 24 25 26 27 28   20 21 22 23 24 25 26   17 18 19 20 21 22 23
29 30                  27 28 29 30 31         24 25 26 27 28 29 30

          Jul                   Aug                   Sep
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
 1  2  3  4  5  6  7                1  2  3  4                    1
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    2  3  4  5  6  7  8
15 16 17 18 19 20 21   12 13 14 15 16 17 18    9 10 11 12 13 14 15
22 23 24 25 26 27 28   19 20 21 22 23 24 25   16 17 18 19 20 21 22
29 30 31               26 27 28 29 30 31      23 24 25 26 27 28 29
                                              30
          Oct                   Nov                   Dec
 S  M Tu  W Th  F  S    S  M Tu  W Th  F  S    S  M Tu  W Th  F  S
    1  2  3  4  5  6                 1  2  3                    1
 7  8  9 10 11 12 13    4  5  6  7  8  9 10    2  3  4  5  6  7  8
14 15 16 17 18 19 20   11 12 13 14 15 16 17    9 10 11 12 13 14 15
21 22 23 24 25 26 27   18 19 20 21 22 23 24   16 17 18 19 20 21 22
28 29 30 31            25 26 27 28 29 30      23 24 25 26 27 28 29
                                              30 31


venus%
```

To construct a calendar for a specific month, type the number of the *month* and the *year*.

Figure 5-4 *Constructing a Monthly Calendar*

```
venus% cal 6 1969
   June 1969
 S  M Tu  W Th  F  S
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30
venus%
```

Type Ahead or Read Ahead

UNIX has an interesting feature called *type ahead* or *read ahead.* Type ahead allows you to type new commands while the system is executing the current command. When you know that you want to type several commands no matter what the outcome of each command, you can type ahead and the system will execute each command in order as it completes the previous command.

Type ahead is most useful when the system requires a lot of time to complete execution of a command. For example, it usually takes the system a bit of time to execute the `lpr` command. You can type `lpr` (followed by RETURN), then `ls`, without waiting for the system to print the command prompt that signals completion of the `lpr` command. This is what it looks like:

Figure 5-5 *Type Ahead or Read Ahead*

```
venus% ls
actors          alaska          hinterland      wilde
venus% lpr wilde
ls
venus% actors          alaska          hinterland      wilde
venus%
```

One side effect of type ahead is that the system types out the results of the `ls` command just after the command prompt that occurs between the `lpr` command and the `ls` command. That looks funny because the RETURN that you pressed after `ls` appears directly after `ls`, not after the command prompt.

Commands as Verbs

As mentioned above, commands are the "verb" of the command line. Some commands have no arguments, many have optional arguments (like the `cal` command's optional month argument), and some require arguments to execute without an error (`lpr` requires a *filename* argument).

Options as Adverbs

An option is one possible argument to a command. The option, sometimes known as a *flag*, tells the command to execute in a certain way. For instance, the `date` command, with its `-u` option, types the Greenwich Mean date and time, rather than the local date and time.

Usually, a `-` precedes the option to a command. `-` is a *marker* for options, and precedes most options on the command line.

Filenames as Objects

Some commands accept filenames as arguments, or "objects," in the command line. `cat`, for example, types out the file contained in its *filename* argument.

Putting the Grammar Together

`ls` is an example of a command with options and filename arguments. When you want to list the files in directory `hinterland`, with the files appearing from most recent file modification to oldest time of modification, use the `-t` option to `ls`:

Figure 5-6 *Command with Option and Filename Argument*

```
venus% ls -t hinterland
north.america   asia
venus% ls hinterland
asia            north.america
venus%
```

Because you created the directory `north.america` after creating the directory `asia`, `north.america` appears first in the file listing when using the `-t` option, and last in the file listing without the option.

To make it absolutely clear, `ls` is the *command*, `-t` is the *option*, and `hinterland` is the *filename* (for a directory) in the above example. Options usually appear before other arguments in the command line.

5.2. Wild Card Characters

At the end of Chapter 4 on files and the file system, you learned abbreviations for the home directory, the working directory, and the parent directory. Sometimes, you can abbreviate filenames too, using certain special characters called *wild card* characters.

The most common wild card characters are: `?` and `*`. `?` is the wild card character representing a one-character value; `*` represents an arbitrary number of characters.

For example, when you want to list all of the files that begin with any two characters and end with a certain *string*, or character combination(in this case `tors`):

Figure 5-7 *Use of ? Wild Card Character*

```
venus% cat > tutors
Socrates
Voltaire
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% ls ??tors
actors  tutors
venus%
```

When you want to list all of the files that contain the string `la`:

Figure 5-8 *Use of * Wild Card Character*

Note: `ls` labels the files it lists in the directory `hinterland` with the directory name, so that there is no confusion about the listing.

```
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% ls *la*
alaska

hinterland:
asia            north.america
venus%
```

These wild card characters are especially useful when you have to search through large directories full of files.

5.3. Redirecting Output

When you created your first file in the section of Chapter 4 on creating files, you used a technique called *redirecting output*, or sometimes *output redirection*, without really knowing what you were doing.

You typed `cat > alice` followed by the quotation you inserted into the file. By typing the > symbol, you "redirected" the output from the `cat` command into the file `alice`.

In other words, when you type > after a command, you redirect the output of that command into the filename after the > symbol. You're telling the system to take what a command would have typed out to the screen and put it into a file instead.

You can also redirect the output of `spell`, a program that checks your spelling. First, create a file full of spelling mistakes with `cat` by redirecting the output of `cat` to place it in the file `mispell`. Then, run `spell` on the file `mispell`,

redirecting the output from `spell` into the file `errors`.

Figure 5-9 *Spelling Correction and Redirecting Output*

```
venus% cat > mispell
"In ordinary composition, use orthodox spelling.  Do not
write nite for night, thru for through, pleez for please,
unless you plan to introduce a complete system of simplified
spelling and are prepared to take the consequences."
venus% spell mispell > errors
venus% cat errors
nite
pleez
thru
venus%
```

`spell` finds the spelling errors in a file and types them in alphabetical order.[25] As in the above example, redirecting the output from `spell` into a file allows you to store the spelling errors until you no longer need to refer to them.

Be careful when redirecting the output of a command to a file. When you redirect output onto a file that already exists, the output redirection will remove the current contents of that file and replace them with the redirected output. You could lose some valuable work.[26]

Figure 5-10 *Caution: Redirecting Output Onto an Existing File*

```
venus% ls
actors          alaska        hinterland      tutors
venus% cat tutors
Socrates
Voltaire
venus% date > tutors
venus% cat tutors
Sat May 30 23:00 GMT 1778
venus%
```

[25] This file is a quotation from *The Elements of Style*, third edition, by William Strunk, Jr., and E.B. White.

[26] To prevent redirected output from "clobbering," or replacing the contents of your files, see the section on the `noclobber` environment variable in *Setting Up Your UNIX Environment: Beginner's Guide*.

Appending Redirected Output

You can avoid "writing over" the contents of a file by *appending* output onto the file, rather than replacing the file. So, if you want to add the date to the end of a file, use the append redirecting output symbol >> .

Figure 5-11 *Appending Output to a File*

```
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% cat actors
James Dean
Humphrey Bogart
venus% date >> actors
venus% cat actors
James Dean
Humphrey Bogart
Mon Sep 14 10:00 PST 1936
venus%
```

When you append output to a file that doesn't exist, the append output symbol, >>, acts exactly like the first redirecting output symbol, >, creating the file and redirecting the command output into it. Try duplicating the example in Figure 5-9 , describing spelling correction and redirecting output, by removing the file `error`, then replacing the redirecting output symbol, > , in the example, with the append output symbol, >> .

5.4. Interactive Programs

So far, the distinction between a *command* and a *program* is blurry. But one kind of program, called an *interactive program*, is never called a command.

An interactive program is a program that expects further instructions after it begins executing.

Note: `bc` may be picky about requiring spaces around arithmetic operators.

`bc`, the basic calculator program, is an interactive program. To begin execution of `bc`, type `bc`. Then, type basic arithmetic expressions, each followed by RETURN, and `bc` will type out the result of the arithmetic calculation. Use any of the following *arithmetic operators*: + for addition, - for subtraction, * for multiplication, and / for division. The RETURN acts as an equals sign. To *quit*, or stop execution of `bc`, type CTRL-D on a line by itself.

Figure 5-12 *Basic Calculations with* `bc`

```
venus% bc
3 + 7
10
3 - 7
-4
3 * 7
21
3 / 7
0 (To quit, type CTRL-D on next line.)
venus%
```

As you can see, `bc` types out results with a certain number of decimal places. When you want `bc` to type results with more decimal places, type `scale =` followed by the *number* of decimal places you want.

Figure 5-13 *Changing Scale in* `bc`

```
venus% bc
3 / 7
0
scale = 5
3 / 7
0.42857 (To quit, type CTRL-D on next line.)
venus%
```

`bc` offers a variety of other mathematical functions, base conversion, and variable assignment and manipulation.[27]

You've learned a lot about commands in this chapter, and you've encountered interactive programs. One of the most frequently used interactive programs on the system is `mail`. See *Mail and Messages: Beginner's Guide* for more information. Another widespread interactive program is `vi`, the UNIX text editor, described in the next chapter.

[27] For information on the other capabilities of `bc`, see *Games, Demos, and Other Pursuits: Beginner's Guide* and the *Commands Reference Manual* entry for `bc`.

6

Editing Files

6

Editing Files

In previous chapters, you created files and sometimes even altered them. To create and make changes to most files, however, it is more convenient to use an interactive program, a *text editor*. The standard UNIX text editor is `vi` (the "visual editor").

6.1. Creating a File

To create a file using `vi`:

- Start `vi`
- Add text to the file
- *Write* the file to save its contents
- Quit, or stop using, `vi`

Starting `vi`

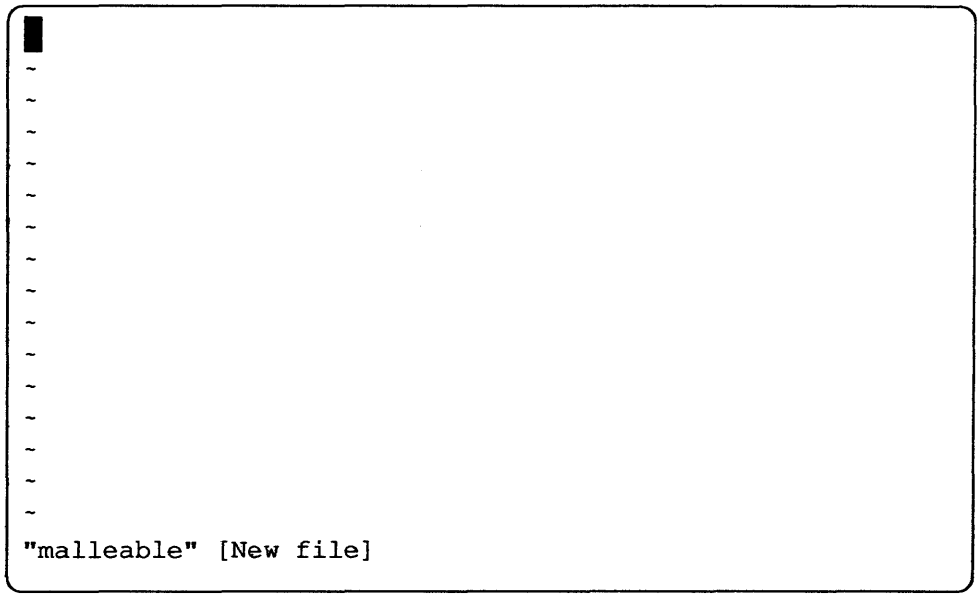
Choose a name for the file you want to create, and type `vi` followed by the *filename*. So, to create the file `malleable`:

Figure 6-1 *Starting* `vi` *with a filename*

```
venus% vi malleable
```

After a short wait, `vi` erases the screen and types out its own interactive screen.

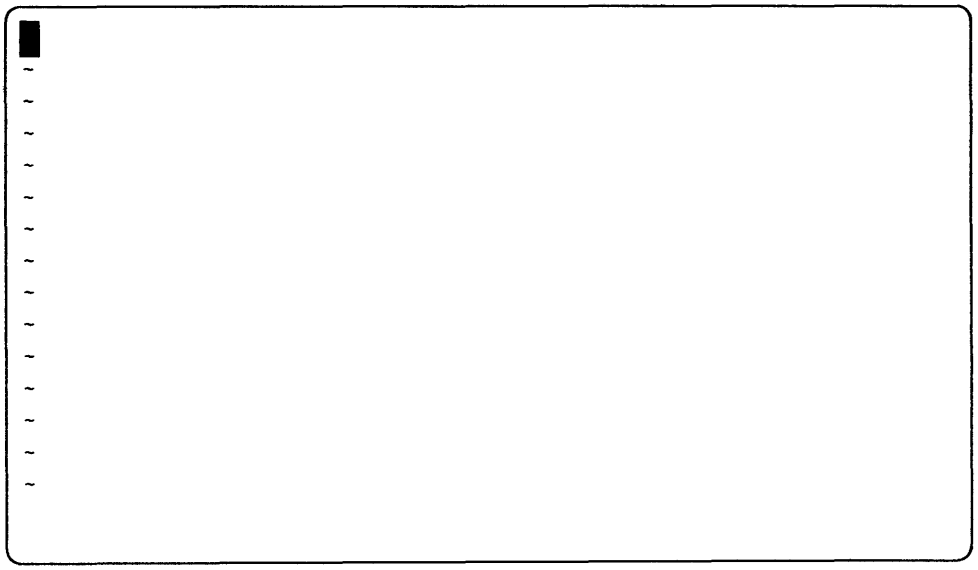
Figure 6-2 `vi` *Interactive Screen*



The last line of the interactive screen shows the filename and indicates that you are creating a new file. It is called the *status line*.

You can also start `vi` without a *filename* by typing just `vi`. In this case, the interactive screen looks like this:

Figure 6-3 `vi` *Interactive Screen Without filename*



Adding Text

To add text to the file, type `a`, for append or add, followed by the text you want to put into the file. The `a` won't show up on the screen or become part of the file; it is an instruction to `vi`.

When you type `a`, you switch from `vi`'s *command mode* to the *insert mode*, allowing you to enter text into the file. You can type as many lines of text as you want. Type [RETURN] at the end of each line of text.

When you're finished entering text into the file, type [ESC] to "escape," or exit from insert mode.

Note: You must explicitly exit insert mode with [ESC], otherwise `vi` assumes you are still attempting to enter text.

Figure 6-4 *Adding Text in* `vi`

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen. (Type [ESC] to exit insert mode.)
~
~
~
~
~
~
~
~
~
~
"malleable" [New file]
```

Writing a File: Saving Your Work

Once you finish adding text to a file, you can save your work, so that when you access the file again, the text you added will still be there.

To save your work, type a colon (`:`) to enter `vi` *colon mode*, then type `w`, to "write" the file.[28]

Note: In contrast to insert mode, you don't explicitly exit `vi` colon mode. Rather, you exit colon mode automatically when you type [RETURN].

[28] Make sure that you exit insert mode by typing [ESC] before typing a colon to enter colon mode.

Figure 6-5 *Writing (or Saving) a File in* `vi`

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
:w
```

After you type RETURN, `vi` confirms that it has written the file by typing the filename, the number of lines in the file, and the number of characters in the file.

Figure 6-6 `vi` *Confirms Writing a File*

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" [New file] 4 lines, 99 characters
```

Quitting `vi`

Now that you've written the file, you can safely quit `vi`. Type `:` to enter colon mode, and `q` to quit the `vi` program.

Figure 6-7 *Quitting* `vi`

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
:q
venus%
```

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

When you decide to quit `vi` without writing a file to which you have added or changed text, `vi` will refuse to let you quit. To demonstrate, add some more text to the example file `malleable`, then try to quit without writing the file.

Figure 6-8 *Caution: Trying to Quit* `vi` *Without Saving Changes*

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.

Add some more text,
but don't write the file. (Type [ESC] to exit insert mode.)
~
~
~
~
~
~
~
~
No write since last change (:quit! overrides)
```

When you really do want to quit without saving your changes, type `:` to enter colon mode and `q!` (an abbreviation for `quit!` that `vi` indicated in the message above). When you do this, `vi` will quit without saving your changes; the next time you access the file, the changes won't appear.

6.2. Moving Around Within a File

Instead of typing an entire file over again when you save it with some mistakes, you can *edit* portions of the file to correct errors, or to make additions or deletions.

However, to access different parts of the file, you have to be able to move around within it. Four commands, for each of four possible directions, allow you to move around a file. They are:

`h` to move left, or "west," one character on the screen

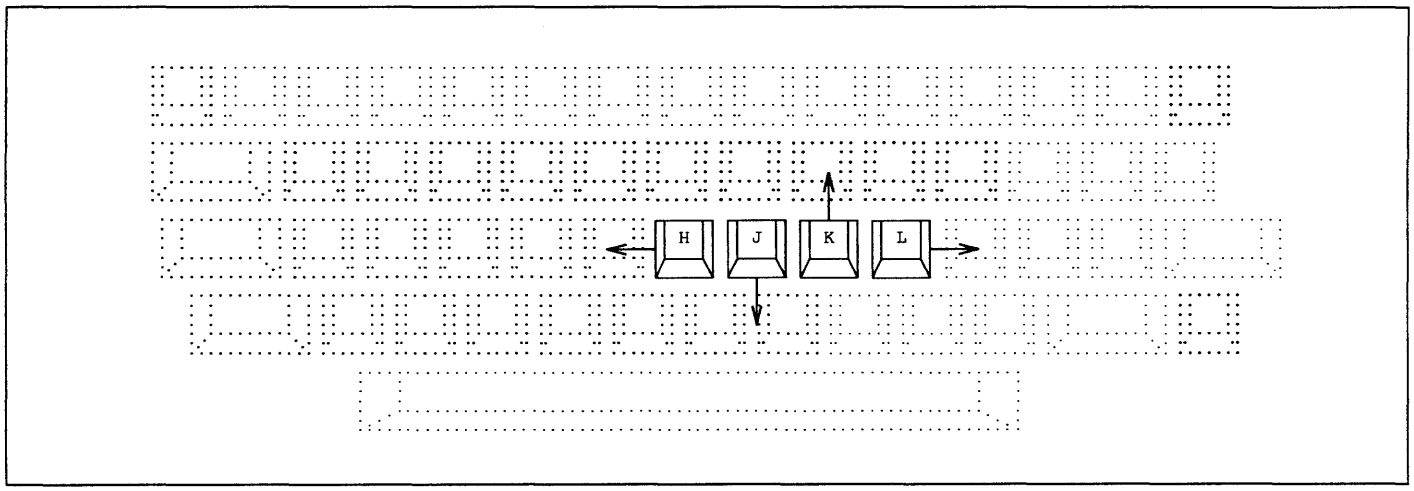
`j` to move down, or "south," one line on the screen

`k` to move up, or "north," one line on the screen

`l` to move right, or "east," one character on the screen

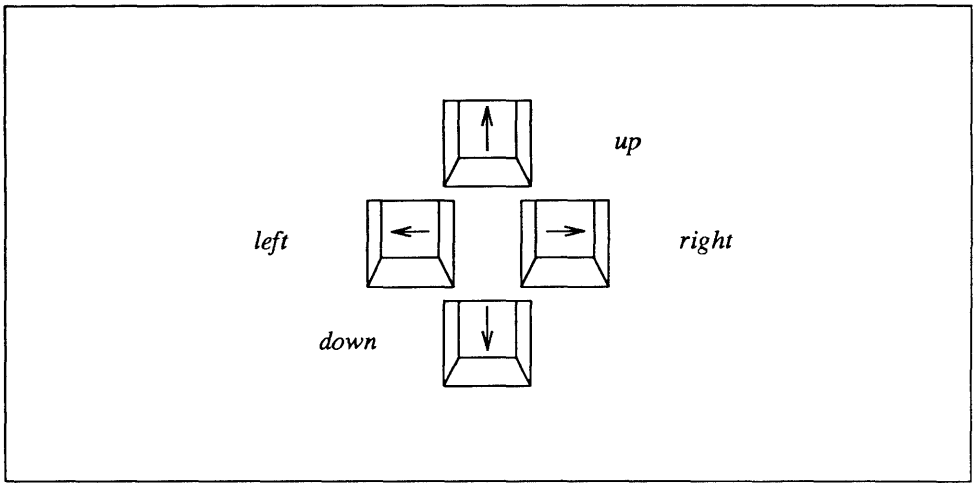
These keys are located in a row on the keyboard.

Figure 6-9 *Cursor Moving Keys for* `vi`



When you have a keyboard that has arrow keys, pointing left, right, up, and down, you may be able to use the arrow keys to move around in the file, in addition to the `h`, `l`, `k`, and `j` keys. Arrow keys looks something like this:

Figure 6-10 *Arrow Keys*



You can move around the file only when you are in command mode. Once again, command mode is the mode you are in when you start `vi`, and when you are in neither text nor colon mode.

Now, type `vi malleable` to work on the file created earlier in this chapter. This time, the `vi` status line won't specify `[New file]`, because `malleable` isn't a new file anymore. You should see this on your screen:

Figure 6-11 *Starting* `vi` *with an Old File*

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

Notice that the cursor, indicating your current position in the file, is just over the letter `A`, the first character in the file `malleable`. When you type `h`, the system flashes the screen (or beeps) and refuses to move the cursor. That is because it can't move the cursor to the left any further.

When you type `j` though, the cursor moves *down* one line onto `t`, the first letter of the next line.

Figure 6-12 *Moving Down in* `vi`

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

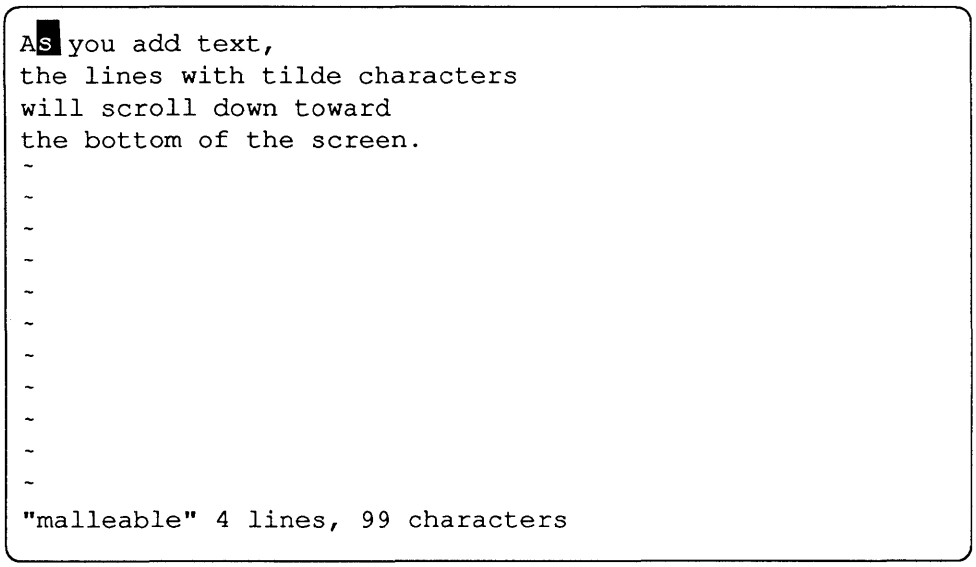
To move *right* one character from the letter `t` to the letter `h`, type `l`.

Figure 6-13 *Moving Right in* `vi`

```
As you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

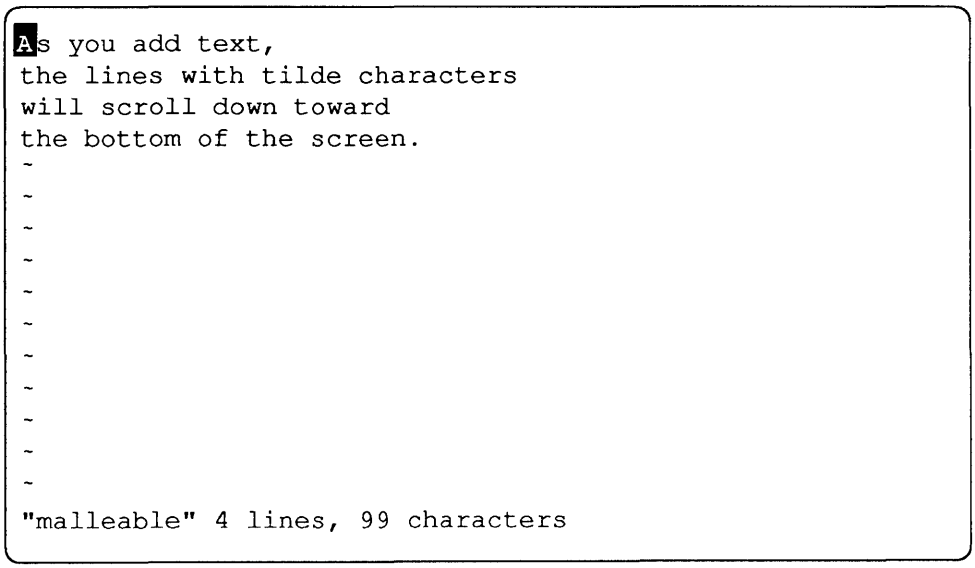
To move *up* one line from the letter `h` to the letter `s`, type `k`.

Figure 6-14 *Moving Up in* `vi`



Finally, to move *left* one character from the letter `s` to the letter `A`, type `h`.

Figure 6-15 *Moving Left in* `vi`



Now, you can move in all four directions within a file. Experiment by moving around to various points in the file.

6.3. Editing a File

There are a few different ways you can edit a file. You can *change* one portion of text to another character string, and *insert* text into or *delete* text from a certain place in a file.

Changing Text

Three basic commands allow you to change text in the file:

`s` to substitute a string for a the character under the cursor

`cw` to change a word, substituting a string for the portion of a word under and after the cursor

`cc` to change an entire line, substituting it with a string that you type

You can access these commands from `vi` command mode; while using them, you are in insert mode. For each command, you exit insert mode by typing [ESC] when you are finished making changes.

Substituting a String

To substitute a string for the character under the cursor, type `s`. The screen will look like the figure below. Then, type in the string with which you want to replace the `A` character, say `As soon a`, followed by [ESC] to exit from insert mode.

Figure 6-16 *Changing Text in* `vi`*: During the* `s` *Command*

```
$s you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

The end result looks like this:

Figure 6-17 *Changing Text in* `vi`*: After the* `s` *Command*

```
As soon as you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

Changing a Word

The `cw` works the same way, except the substitution symbol `$` appears on the last character of the word you are changing. Type `h` four times to move left onto letter `o`, the second character in the word `soon`. Then, type `cw`.

Figure 6-18 *Changing Text in* `vi`*: During the* `cw` *Command*

```
As soo$ as you add text,
the lines with tilde characters
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

You can type in the text you want to substitute for the rest of the word, say `lowly`, followed by [ESC] to exit insert mode. You can see the changes made to the first line in the next figure, illustrating the `cc` command.

Changing a Whole Line

The `cc` command is similar to `s` and `cw`, except that it replaces an entire line of text, and it doesn't use the substitution symbol `$`. Move to the second line of text by typing `j`, and to the beginning of that line by typing `h` repeatedly. Then type `cc` to substitute for an entire line of text.

Figure 6-19 *Changing Text in* `vi`*: During the* `cc` *Command*

```
As slowly as you add text,
will scroll down toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

Type the string that you chose as a replacement to the deleted line, say `the subsequent text`, then type [ESC] to exit from insert mode.

Inserting Text

You can insert text with three basic commands:

`i` to insert text under the cursor

`o` (lower-case letter o)
to insert text on a new line created just after the current line

`O` (upper-case letter O)
to insert text on a new line created just before the current line

Just as with the changing text commands, you access these commands from `vi` command mode. While using them, you are in insert mode. For each command, you exit insert mode by typing [ESC] when you are finished making changes.

The next example shows the `i` command, but you should also try the other inserting text commands: `o` and `O`.

Move to the space between `down` and `toward` using a combination of `h` and `j` commands. Type `i`, a space character, then the text `quickly` under the cursor. Then type [ESC] to exit from insert mode. You should see something like this:

Figure 6-20 *Inserting Text in* `vi`*: The* `i` *Command*

```
As slowly as you add text,
the subsequent text
will scroll down quickly toward
the bottom of the screen.
~
~
~
~
~
~
~
~
~
~
~
"malleable" 4 lines, 99 characters
```

Deleting Text

The three basic commands for deleting text differ from the text changing and inserting commands because they don't use insert mode. You are in command mode the entire time when you are deleting text.

The three commands are:

`x` to delete the character under the cursor

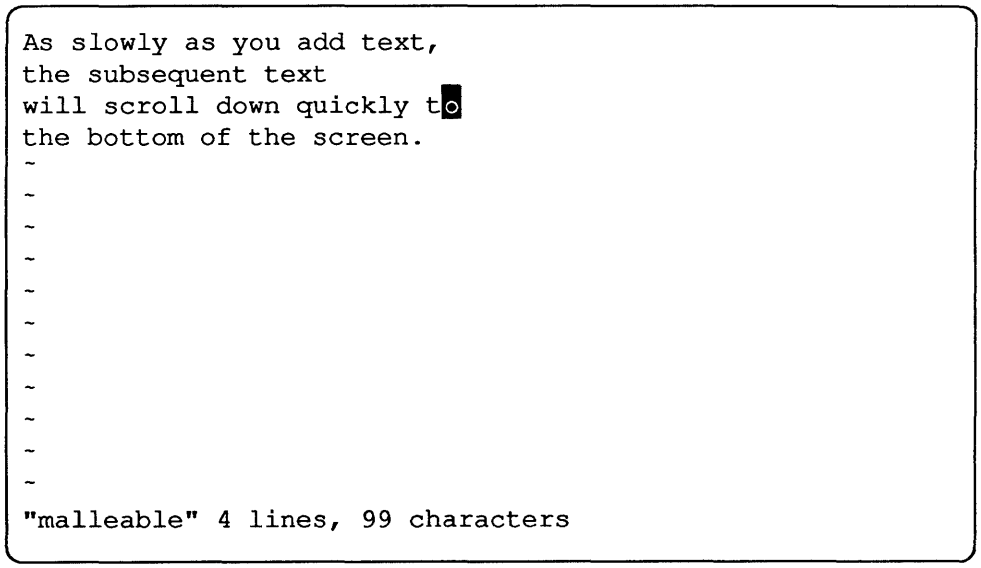
`dw` to delete the portion of a word under and after the cursor (including the space after the word)

`dd` to delete the entire line the cursor is on

The following example shows the `x` command, but you should try the other deleting text commands, `dw` and `dd`, on your own.

Move to the letter `w` in the word `toward`. Type `x` four times, deleting the string `ward`. Don't type ESC; you aren't in insert mode, because deleting text commands use base mode.

Figure 6-21 *Deleting Text in* `vi`*: The* `x` *Command*



When you delete a line of text with `dd`, `vi` erases the status line from the screen, until you write the file again.

6.4. Where to Find Out More About Editing Files

Now you know the basics of `vi`, you can find out about the many other useful `vi` commands, by looking at *Editing Text Files on the Sun Workstation.*

7

Formatting Documents

7

Formatting Documents

Sometimes it is a lot of work to create a file and print it out the way you want it to look. A *text formatter* allows you to specify concisely the way you want your printout to look, and to add special attributes, like different font sizes, intensities, and types, that would be impossible to access without the formatter.

In this chapter, you will learn the basics of the UNIX formatter called `nroff`.[29]

7.1. Sample Memo

To learn `nroff`, you can create a sample memo which can act as a template for memos that you want to create in the future. Here is the sample memo, after formatting:

From: Board of Directors
To: Vociferous Employees
Cc: All 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.

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 have terminated the service contract for the printers.

3) We will sell half of the printers by the end of the year.

If these steps do not stop the proliferation of memos, we will consider subtracting the cost of employee memos from their salaries.

[29] UNIX also has a formatter called `troff`, which is equivalent to `nroff`, except that it produces output for typesetters and related printers.

Here is the sample memo before formatting:

```
.LP
From: Board of Directors
.br
To: Vociferous Employees
.br
Cc: All Employees
.sp
.ce
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.
.LP
We have decided to take three immediate steps to counter this
.ul
threat
to corporate profitability:
.IP 1)
We will reduce paper stocked for the printers by 25% within month-end.
.IP 2)
We have terminated the service contract for the printers.
.IP 3)
We will sell half of the printers by the end of the year.
.LP
If these steps do not stop the proliferation of memos, we will
consider subtracting the cost of employee memos from their salaries.
```

To create the file you want formatted, you must learn the basic `nroff` formatting commands. All of the lines that begin with a period character in the listing above are `nroff` formatting commands.

Then, to transform the "raw" `nroff` file into a finished document, you must learn to run the formatter `nroff` on the "raw" file. Finally, you can print out the formatted file.

7.2. Basic Formatting Commands

Use the text editor `vi` to create the file `sample.memo`, with all of the memo text and formatting commands in it.

To construct the "raw" nroff file for formatting, you will need the following `nroff` formatting commands:

`.LP` to *left-justify* a paragraph

`.IP` to created an *itemized* paragraph (like this one)

`.ce` to center text on the page

`.ul` to underline portions of text

`.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 number one, `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, like the `nroff` commands listed at the beginning of this section.

Centering Text

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

Underlining Text

To underline text, type the `.ul` command on a line by itself, followed by the text you want underlined on the next line. Start a new line after the text you want underlined, because all of the text on the line following the `.ul` command will be underlined.

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, rather than *filling* in all of the text until the next paragraph on the subsequent lines, type `.br` on a line by itself, just after the line of text that you want to end at a specific point.

7.3. Running the Formatter

Now, you should have a file in one of your directories called `sample.memo`. It should contain all of the memo text and formatting commands, and it should look like the sample file, before formatting, that you saw near the beginning of this chapter.

To run the `nroff` formatter on the file, formatting the text contained in it, type `nroff -ms`, followed by the *filename* of the file that is ready to be formatted, the redirecting output symbol >, and the *filename* you want for the file after it is formatted and ready to go to the printer.

Note: The option `-ms` means "run the formatter with the *macro package* `ms`." A macro package is a set of formatting command definitions associated with the `nroff` formatter.

For the file `sample.memo`, type:

Figure 7-1 *Running the* `nroff` *Formatter*

```
venus% ls
sample.memo (along with the other files in that directory)
venus% nroff -ms sample.memo > sample.ms
venus%
```

Often, people give their formatted files a filename with the last part, or *filename extension*, of the macro package they used to format the file.

It may take a little while for the `nroff` to finish formatting the file.

7.4. Printing the Memo

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

Figure 7-2 *Printing the Formatted File*

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

7.5. Where to Find Out More About Formatting

Since you now know the basic `nroff` commands, you may want to find out more. Look at *Using nroff and troff on the Sun Workstation* and *Formatting Documents on the Sun Workstation.*

8

Searching Through Files

8

Searching Through Files

Many people find that they want to search for a certain string of text within a file, but they would prefer not to look at the entire file to find it.

`grep` is the answer to their worries. `grep`, or the **g**lobal **r**egular **e**xpression **p**rinter, searches through a file or files for the string, or *expression*, that you specify. This chapter will help you learn how to use `grep` effectively.

Note: The term *regular expression* has a special meaning in UNIX. You will learn more about regular expressions in *Doing More With UNIX: Beginner's Guide.*

8.1. Basic Searches with `grep`

To search for a particular character string in a specified file, use the `grep` command. If you have the file `people` in your working directory, you can search it with `grep`.[30]

Figure 8-1 *File to Search with* `grep`

```
venus% cat people
Emily Dickinson   X7984
Edgar Allen Poe   X9096
Louisa May Alcott X4236
venus%
```

This file isn't long. It would be easy to search it by looking at it, but we're experimenting with the `grep` command.

[30] You can create the file `people` using `vi`, or redirecting the output of `cat`. To look at the file, type `cat people`.

To find out Edgar Allen Poe's telephone extension, type `grep` followed by the portion of his name you want to use to search for him and the *filename* of the file you want to search, in this case `people`.

Figure 8-2 *Searching with* `grep`

```
venus% grep Poe people
Edgar Allen Poe   X9096
venus% grep Allen people
Edgar Allen Poe   X9096
venus% grep allen people
venus%
```

You can use any part of Poe's name to search for the line of the file that contains his phone extension. However, `grep` pays close attention to the case of the characters you want it to search for. That's why it didn't find anything when it searched for `allen`, beginning with a lower-case letter.

Sometimes, `grep` will find more than one occurrence of the search string. When you want to find the phone extensions of all people with names that have the string `Al` in them:

Figure 8-3 `grep` Finds More Than One String

```
venus% cat people
Emily Dickinson   X7984
Edgar Allen Poe   X9096
Louisa May Alcott X4236
venus% grep Al people
Edgar Allen Poe   X9096
Louisa May Alcott X4236
venus%
```

`grep` found the string `Al` in the entry for Poe and for Alcott.

8.2. `grep` with Multi-Word Strings

When you try to `grep` for a string that is more than one word long, you will encounter a problem. For example, look for `Louisa May` in the `people` file.

Figure 8-4 *Caution: Trying to* `grep` *for a Multi-Word String*

```
venus% grep Louisa May people
May: No such file or directory
people:Louisa May Alcott X4236
venus%
```

`grep` interpreted the instruction so as to search for the string `Louisa`, in the files `May` and `people`, rather than searching for the string `Louisa May` in the file `people`. That's why `grep` typed out the error indicating that the file `May` doesn't exist, then specified that the file `people` does indeed contain the string `Louisa` on the line giving Louisa May Alcott's phone extension.

The way to prevent this confusion is to enclose all multi-word search strings with double quotes. Many UNIX commands require quotes around multi-word arguments.[31]

Figure 8-5 *Properly Quoted* `grep` *Search String*

```
venus% grep "Louisa May" people
Louisa May Alcott X4236
venus%
```

8.3. Searching More Than One File

In the example that needed double quotes above, you may have noticed that `grep` attempted to search for a string in two files. In fact, `grep` can search for a string in groups of files.

When you have a whole directory of files, and you want to search for a specific string in all of them, type `grep` followed by `*`, the wild card character for a string of arbitrary length. For this example, return to the home directory you created in Chapter 4 on files and the file system and modified in Chapter 5 on UNIX commands.

Figure 8-6 *Searching A Directory of Files*

```
venus% pwd
/usr/medici
venus% ls
actors          alaska        hinterland      tutors
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%
```

`grep` does not attempt to search directories, like `hinterland`, or their contents.

[31] See *Doing More With UNIX: Beginner's Guide* for more information about quotes and command arguments.

8.4. Searching for Lines Without a Certain String

Sometimes, you may want to search for all the lines of a file that **don't** contain a certain string.

To do this, 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`:

Figure 8-7 *Searching for Lines That Don't Contain a String*

```
venus% ls
actors       alaska     hinterland   tutors       wilde
venus% grep -v e *
actors:Mon Sep 14 10:00 PST 1936
wilde:That is all.
venus%
```

8.5. Where to Find Out More About `grep`

You have learned `grep` well, but when you want to become a `grep` expert, you can find out more in *Using UNIX Text Utilities on the Sun Workstation* and in the *Commands Reference Manual.*

9

Timesaving Features

9

Timesaving Features

Timesaving features on the UNIX system can make your work quicker and easier. In this chapter, you will learn about *aliases* and basic *history* command repetition and substitution, shorthand ways of typing many commands. In addition, you will learn about running commands in the *background*, a procedure where you can instruct the system to execute multiple command lines at the same time.

9.1. Aliases

You can use the `alias` command to develop a shorthand for commands that you type frequently. Since you probably type the `logout` command every time you end a work session, it is a good example of a command you might want to abbreviate.

To alias a command, type `alias` followed by the *alias*, then the command *string* you wish to alias. So, to abbreviate the command `logout` as `lo`:

Figure 9-1 *The* `alias` *Command*

```
venus% alias lo logout
venus% lo

venus login:
```

When you type the `lo` alias for the `logout` command, the system interprets it as if you had typed `logout`, and logs you out.

Whenever you want to create an alias for a multi-word command string, remember to enclose the multi-word string in double quotes. So, if you want to abbreviate `grep -v` as `gv`:

Figure 9-2 *Aliasing a Multi-Word Command String*

```
venus% alias gv "grep -v"
venus% pwd
/usr/medici
venus% gv e *
actors:Mon Sep 14 10:00 PST 1936
wilde:That is all.
venus%
```

To learn more about aliases, see *Setting Up Your UNIX Environment: Beginner's Guide.*

9.2. The History Mechanism

The UNIX system stores the commands you type using a *history mechanism.* You can reuse the commands you have typed with *command repetition* and *command substitution.*

Command Repetition

Command repetition allows you to repeat previous command lines or parts of previous command lines. The two most common command repetition commands are: `!!` and `!$`.

`!!` repeats the entire last command line you entered. For example, when you create a file and make a mistake, you may want to start over, repeating the original command line.

Figure 9-3 *Entire Command Line Repetition with* `!!`

```
venus% cat > mammals
dog
cat
chimp
zbra (Mistake occurs here; type CTRL-D to save file.)
venus% !!
cat > mammals
dog
cat
chimp
zebra (Save correct file with CTRL-D)
venus%
```

The system types out the command line repeated by `!!` so that you can make sure it is the right one.

When you just want to repeat the last word on the previous command line, type `!$`. You can type both `!!` and `!$` anywhere in the current command line to repeat all or a portion of the previous line.

For example, when you cat the `mammals` file you just created, you may want to print it out afterward.

Figure 9-4 *Last Word of Command Repetition with* `!$`

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

Once again, the system types out the command line you are executing, so you can make sure it is right.

Command Substitution

Sometimes you may want to repeat a previous command line, but change a portion of it, perhaps to correct a mistake or to complete a repetitive task. *Command substitution*, a basic form of *command editing*, is useful for such corrections and tasks.

For example, when you want to make a directory called `animals` (perhaps so that you can move `mammals` into it), you should type `mkdir animals`. But, if you make a mistake, you can correct your mistake with the command substitution symbols, `^ ^` , the caret characters.

Figure 9-5 *Correcting Mistakes with Command Substitution*

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

You successfully correct the error with command substitution, and the system makes the directory `animals`.

To use command substitution for repetitive tasks, consider that you may want to make five directories, for the five known kingdoms of life. You've already created the `animals` directory, but here is an easy way to create the rest of them.

Figure 9-6 *Command Substitution to Aid Repetitive Tasks*

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

To learn more about the history mechanism, and *command line editing*, see *Doing More With UNIX: Beginner's Guide.*

9.3. Running Commands in the Background

Another way to save your time is by running commands in the *background*, one aspect of *job control.* In fact, you can run several commands in the background at once. When you run commands in the background, you can run another command, as usual, running it in what is called the *foreground.*

To run a command in the background, type the ampersand character, `&` , at the end of the command line. For example, to run `spell` on the file `mammals` in the background:

Figure 9-7 *Running a Command in the Background*

```
venus% spell mammals > mammals.error &
[1] 69
venus% ls
animals         mammals         monerans        protistans
fungi           mammals.error   plants
[1]  + Done                  spell mammals > mammals.error
venus% cat mammals.error
venus%
```

The system works on commands you run in the foreground, `ls` in this example, *at the same time* as working on the command you ran in the background, `spell`.

When you list files in a directory where a command has started to type output into a file, the listing will contain the new file, even though the file is not yet complete.

In this case, `mammals.error` is an empty file, because there are no spelling errors in the file `mammals`.

The system won't type the message indicating it has finished the background job until:

1) it has finished execution of the background command

2) you ask it to perform another command

To learn more about background processing and *job control*, see *Doing More With UNIX: Beginner's Guide*. Since you have nearly completed the basic UNIX tutorial, be aware that *Doing More With UNIX* is the next manual that covers UNIX commands within the *Beginner's Guide* series.

10

Online Documentation

10

Online Documentation

When you want quick access to documentation, or when you don't have the hard-copy manuals around, try using the *online* documentation. The primary online documentation source is the *Man Page*, short for "manual page." Other online documents, and tutorials associated with programs, may help to meet your documentation needs too.

10.1. Man Pages

Man Pages exist for just about every command you can type to the command prompt (and some that you can't). Man Pages give a basic outline of the command, including the following sections:

Name
: Name and function.

Synopsis
: Syntax diagram. Name, followed by options (in brackets), and any further arguments, like *filenames*.

Description
: Brief description of the command.

Return Value
: Only in **System Call** Man Pages. Lists the value returned with successful and unsuccessful operations.

Errors
: Only in **System Call** Man Pages. Lists possible errors.

Options
: Usually, and itemized list of options in convenient order, with a brief description of each option.

Commands
: In the case of an interactive program, commands that you can type during program execution.

Files
: Files related to the command.

See Also
: Related commands and pertinent documentation.

Diagnostics
List of diagnostic messages that may be produced.

Bugs
Known bugs, or problems with the command.

To access the Man Pages, type `man` followed by the *command name*.[32]

`ls`: An Example

For example, to access the Man Page for `ls`:

Figure 10-1 *Accessing an Online Man Page*

```
venus% man ls

LS(1)                        USER COMMANDS                        LS(1)


NAME
     ls - list contents of directory

SYNOPSIS
     ls [ -acdfgilqrstu1ACLFR ] name . . .

DESCRIPTION
     For each name which is a directory, ls lists the contents of
     the directory; for each name which is a file, ls repeats its
     name and any other information requested. By default, the
     output is sorted alphabetically. When no argument is given,
     the current directory is listed. When several arguments are
     given, the arguments are first sorted appropriately, but
     file arguments are processed before directories and their
     contents.

OPTIONS
     There are a large number of options:

     -a   List all entries; in the absence of this option,
          entries whose names begin with a period (.) are not
          listed.

     -c   Use time of file creation for sorting or printing.

     -d   If argument is a directory, list only its name; often
          used with -l to get the status of a directory.

     -f   Force each argument to be interpreted as a directory
--More--(20%)  (Type q to quit, space to continue.)
```

[32] When you don't know the name of the command you want to access, just its functions, try the `-k` option of the `man` command. See *Doing More With UNIX: Beginner's Guide* for more information on `man -k`.

`man` uses `more` to type the Man Page on the screen.

Using an Option to `ls`

When you read the `ls` Man Page, you may come across the `-F` (capital `F`) option to `ls`. This is a useful option to know about, the kind of discovery you can make when you read Man Pages for commands you already know about.

`ls -F` lists the contents of a directory like `ls`, except that it appends a slash character, `/`, to each directory name to distinguish directories from files.[33]

List (with the `-F` option) the directory with all the life forms in it.

Figure 10-2 *Listing a Directory with* `ls -F`

```
venus% ls -F
animals/        mammals         monerans/       protistans/
fungi/          mammals.error   plants/
venus% ls
animals         mammals         monerans        protistans
fungi           mammals.error   plants
venus%
```

Congratulations — you've read the main portion of this manual and learned the basics of UNIX. You may find the appendices useful too.

[33] The `-F` option to `ls` also distinguished *symbolic links* from directories and files, but don't worry about this for now.

A

Further Reading

A

Further Reading

You've learned a lot! When you want to read more, start with these manuals:

Setting Up Your UNIX Environment: Beginner's Guide
Self Help With Problems: Beginner's Guide
Windows and Window-Based Tools: Beginner's Guide
Mail and Messages: Beginner's Guide
Games, Demos, and Other Pursuits: Beginner's Guide
Doing More With UNIX: Beginner's Guide
Using the Network: Beginner's Guide

Editing Text Files on the Sun Workstation
Formatting Documents on the Sun Workstation
Using `nroff` *and* `troff` *on the Sun Workstation*
Using UNIX Text Utilities on the Sun Workstation
Commands Reference Manual
System Administration for the Sun Workstation

B

Command Summary

B

Command Summary

This is a summary of all commands mentioned in this manual. Each command appears in alphabetical order by name, and includes a syntax diagram, and a brief paragraph describing its function. Separate sections list **Wild Card Characters, Redirecting Output Symbols, History Mechanism Commands,** and the **Job Control Command** covered in this manual.

For information on control key usage, see the section on control keys, Section 2.5 .

B.1. Basic Commands

`alias` `alias` *aliased-command unaliased-command*
assigns the alias *aliased-command* to the command string *unaliased-command.*

`bc` `bc`
is the interactive basic calculator program that can handle arithmetic functions like addition, subtraction, multiplication, and division. Type RETURN after each arithmetic expression to get `bc` to evaluate it. Set the number of decimal places with `scale =` followed by the number of decimal places you desire in calculation results.

`cal` `cal` [*numeric-month*] *year*
types out a calendar of a given year, or optionally of a certain month of a year.

`cat` `cat` *filename* `. . .`
types a file or files to the screen, or redirects output (`cat` *filename* `. . .` > *filename*), into another file.

`cd` `cd` [*directory-name*]
or change directory, changes the working directory to the home directory, or optionally to the *directory-name* destination.

`cp` `cp` *source-filename destination-filename* or
`cp` *source-filename destination-directory-name*
copies a file into a filename, or into a directory. Watch out for existing files.

`date` — `date [-u]`
types out the current date and time in the local time standard. With the `-u` option, `date` types out universal, or Greenwich Mean, time.

`grep` — `grep [-v]` *search-string filename . . .*
searches for the specified *search-string* in the specified file or files, and types out all text lines containing the *search-string*. With the `-v` option, `grep` types out all text lines that **don't** contain the *search-string*.

`logout` — `logout`
logs out of the system.

`lpr` — `lpr` *filename . . .*
prints out a copy of the file or files on the printer.

`ls` — `ls [-tF]` *filename . . .* or
`ls [-tF]` *directory-name . . .*
types out a listing of the specified files, or of the files and directories in the specified directory. The `-t` option types the listing in reverse chronological order by last time modified, and the `-F` option distinguished directories by appending the slash symbol onto them.

`mail` — See *Mail and Messages: Beginner's Guide*

`man` — `man` *command-name . . .*
types out the Man Page, or manual page, for the specified command, giving information about syntax, options, etc., for that command.

`mkdir` — `mkdir` *directory-name . . .*
makes a directory with name *directory-name*.

`more` — `more` *filename . . .*
prints out the file *filename* page-by-page. `more` is especially useful for long files. Use `q` to quit, the space bar to continue paging.

`mv` — `mv` *source-filename destination-filename* or
`mv` *source-filename destination-directory-name* or
`mv` *source-directory-name destination-directory-name*
moves, or renames, a file to another file, a file to a another directory, or a directory to another directory.

`nroff` — `nroff -ms` *filename . . .*
formats the specified file. Often, you will want to redirect the output of `nroff` to a file.

Use any of the following formatting commands in the source file:

`.LP` to l*eft-justify* a paragraph

`.IP` to created an i*temized* paragraph (like this one)

`.ce` to center text on the page

`.ul` to underline portions of text

`.sp` to create a blank line space

`.br` to force the end of a line, a line break

`passwd` `passwd`
changes password by prompting for old password and twice for new password.

`pwd` `pwd`
types working directory name.

`rm` `rm [-r]` *filename* . . . or
`rm [-r]` *directory-name* . . .
removes, or deletes, specified files or directories. With `-r` option, removes recursively the contents of any directories, in effect deleting all children of that directory.

`rmdir` `rmdir` *directory-name* . . .
removes dirRectories specified, when they are empty directories.

`spell` `spell` *filename* . . .
checks spelling of words in specified file and types them out to screen, unless you redirect the output to a file.

`troff` See *Using nroff and troff on the Sun Workstation.*

`vi` `vi` *filename* . . .
text editor with the following commands (in command mode, unless preceded by a `:`):

`a` add text

`cc` substitute a line with a string (enters insert mode)

`cw` substitute, or change, a word with a string (enters insert mode)

`dd` delete the entire line the cursor is on

`dw` delete the word, or portion of word, under and after the cursor

`h` move left, or "west," one character

`i` insert text under the cursor (enters insert mode)

`j` move down, or "south," one line

`k` move up, or "north," one line

`l` move right, or "east," one character

`o` insert text on a new blank line after the current line (enters insert mode)

`O` insert text on a new blank line before the current line (enters insert mode)

`s`	substitute a character with a string (enters insert mode)
`x`	delete the character under the cursor
`:q`	quit `vi`
`:q!`	quit `vi`, without writing changes
`:w`	save, or write a file

B.2. Wild Card Characters

`?`	Wild card representing a one-character value in a filename.
`*`	Wild card representing an arbitrary number of characters in a filename.

B.3. Redirecting Output Symbols

`>`	Symbol representing redirecting output from a command to a filename. Watch out because `>` may write over the destination file.
`>>`	Symbol representing **appending** output from a command to a filename. `>>` will append onto, not write over, the destination file.

B.4. History Mechanism Commands

`!!`	Repeats entire last command line at any point in current command line.
`!$`	Repeats last word of last command line at any point in current command line.
`^^`	Substitutes portion of previous command line between symbols for new string after symbols.

B.5. Job Control Command

`&`	Pushes command to background for execution while you are typing other commands in the foreground.

C

Glossary

C

Glossary

This glossary contains a list of UNIX terms in common use, especially in this manual. For commands, see the command summary appendix B , or the "cheat sheet" at the end of this manual.

absolute pathname
The list of directories starting with the root directory, specified as /, down through the file system tree structure to the file or directory in question, with each directory along the way separated from the others with an additional / character. The other types of pathnames are: simple and relative pathnames.

account
The means by which a user access the system and the system administrator assigns space for the user's files and directories. Usernames and passwords are specific to accounts.

alias
An alias is a user-specified abbreviation, or alternate string, for a standard command string.

append
Attach to the end.

append output
Attach redirected output to the end of a file.

argument
Any word (string of characters separated by spaces or tabs) occurring after the command in a command line.

arithmetic operator
Symbols used to indicate and execute addition (+), subtraction (−), multiplication (*), and division (/).

background
The place where you can run a command, or commands, and still type commands in the foreground, with background and foreground commands executing apparently simultaneously.

bug
Any problem or error in the design or coding of a program.

case-sensitive
Treating lower-case and upper-case characters as two kinds of characters with separate functions.

child directory
The directory directly below "this" directory in the file system tree structure.

colon mode
In `vi`, the mode where you can write files or quit the program. Type `:` when in command mode to access colon mode. Besides command mode and colon mode, `vi` has insert mode.

command
A string of characters that one types to the system, expecting it to respond by performing a certain function unique to that command line. The command is sometimes called the "verb" of the command line "sentence."

command editing
Modifying a previous command line for reuse as a new command.

command line
A string of characters beginning with a command followed by arguments, which aren't necessarily required, including options, filenames, and other expressions.

command mode
In `vi`, the mode where you can type moving and deleting commands, as well as changing text commands that access insert mode. `vi` starts out in command mode; the other modes are insert mode and colon mode.

command prompt
The string of characters that the system types telling you it is ready to accept and interpret your next command line. Often, the command prompt includes the name of the system.

command repetition
Repeating a previous command line, or portion of one, for reuse as a new command line.

command substitution
Substituting a portion of a previous command line for reuse as a new command line.

concatenate
Literally, to link together in series. For UNIX, to type a file on the screen.

console
When you log in at the actual machine, rather than accessing the machine from another machine, you log in on the console.

control keys
Keys that require that you press and hold down the CTRL key while typing the associated character to perform a certain function. For example, CTRL-U deletes the current command line before execution. Control keys

are a way of extending the functionality of the keyboard.

current directory
See **working directory.**

cursor
The rectangular portion of the screen that moves as you type keys on the keyboard indicating your current position on the screen.

debug
To attempt to fix problems with programs.

directory
A "container" for files and other directories that resides within the UNIX file system in a tree structure. You can make, move, copy, and remove directories and their associated files. Really, directories are a special type of file.

downtime
Time when the computer is not working because of maintenance or an unknown problem.

echo
The way that the system types back the keys that you type onto the screen.

editor:
See **text editor.**

empty directory
A directory that doesn't contain any files or directories.

end-of-file character
CTRL-D the character used to indicate when you have finished typing in a file.

error message
A character string that the system types to let you know that there is a problem with a command, or perhaps with something else. Usually, the error message will give you some idea of how to correct the problem.

escape keys
Keys that require that you press and release the ESC key before typing the associated character. Escape keys are a way of extending the functionality of the keyboard.

expression
An expression is a string of characters that signifies a certain meaning to the system.

file A "container" for text. You can create, move, copy, list, and remove, or delete files. They are a good way to save memos, phone lists, programs, and other portions of text. You can use a text editor to create and modify files.

filename
The name that you assign to a file. Try not to use special characters in the filename, because the system may misinterpret the filename as something

else.

filename extension
A portion of a filename appended to the end, often demarcated with a period character. An example is the `.ms` designation at the end of filenames of `nroff`-formatted files.

file protection
The way that the system maintains some security over the contents of your files.

file system
In the case of UNIX, a tree-structured network of files and directories, through which you can move to access the files and directories contained in it.

file system hierarchy
The structure of the file system, consisting of a tree of files and directories, with a root directory at the "top," and directories acting as parent directories and child directories throughout.

filling
The process of placing a series of words on a line until the end of the line, then placing the words on the next line until the end of that line, and so on. The `nroff` formatter fills text unless you type a command, such as the line break command `.br`, indicating that it do otherwise.

foreground
The place where you type commands to the command prompt for the system to execute. You can type an ampersand character at the end of a command line to have the system execute that command line in the background, while you continue to type commands to the command prompt for execution in the foreground.

formatter
See **text formatter.**

Greenwich Mean Time
The time at the Greenwich meridian, by which people establish all of the time zones on earth. Sometimes known as universal time.

hardcopy
A copy of a text file that is on a piece of paper, not stored somewhere on the computer.

hidden files
Files that don't show up when you do a simple listing, because they have a period character as the first character in their filename.

hierarchy
See **file system hierarchy.**

history mechanism
The way that the system keeps track of commands that you have typed previously so that you can reuse them with command repetition and command

editing.

home directory
The directory you are in when you first log in to the system, which is also your personal root directory to all the files and directories you create in your own area.

hostname
The name of a machine, or host.

indent
To put spaces or tabs before a section of text, particularly at the beginning of a paragraph.

insert mode
For `vi`, insert mode is the mode that you enter when inserting or changing text, and exit by typing ESC. The other `vi` modes are: command mode and colon mode.

interactive program
A program that requires you to type commands to it after you have started it, rather than executing its function in full after you enter a single command line.

interrupt
To stop the execution of a command or program.

job control
The way that the system keeps track of all the commands that you run in the foreground and background, as well as all of the commands that other users may run on the system.

keyboard
A part of the Sun workstation, similar to a typewriter, that permits you to input characters that usually appear on the screen.

left-justify
To put characters up against the left margin, or side of the page. Often mentioned with `nroff` formatting, when you may want to start a paragraph against the left side, without any indentation.

line break
A way to stop the flow of filling, so that a line of text ends at a specific point, with any further words appearing on the next line. Often mentioned with `nroff` formatting, when you want to stop filling a particular section of text.

line space
A way to indicate a blank vertical spacing equivalent to one line, usually mentioned when formatting with `nroff`.

login
Gaining access to the system, usually by typing a username and password, so that you can begin a work session on the computer.

login prompt
The string of characters that the system types to let you know that it is ready to interpret your username when you decide to type it.

logout
Ending access to the system, usually when you finish your work and you don't want a random person to have access to your account, perhaps causing damage to one of your files, or accessing sensitive information.

Man Page
An online manual page, accessible with the `man` command, that provides information on most UNIX commands.

marker
In this case, the - character is marker for options, so that when you type a command with an option, the system can figure out that you have typed an option, not another kind of command argument.

message of the day
A portion of text that the system may type when you first log in. The system administrator sometimes creates this message to let users know about down-time or other important system events.

mode
See **command mode**, **colon mode**, or **insert mode**.

mouse
A small, rectangular part of the Sun workstation, with three buttons and a wire running out of it, that permits better control of the window system, when it is running. The mouse usually doesn't have fur.

mouse tablet
A shiny, flat, rectangular part of the Sun workstation that permits you to move around your mouse in a way that the system can understand.

natural language
The languages that people, as opposed to machines, speak, read, or write in societies and cultures of all eras.

online documentation
The system of Man Pages that permits you to access information about commands while you are logged in to the system.

operating system
A collection of programs that monitors the use of the machine and supervises the other programs executed by it.

operator
See **arithmetic operator**.

option
A portion of the command line sometimes compared with an adverb because it modifies the effect of the command you type.

output redirection
See **redirecting output.**

Pacific Standard Time
The standard for time on the west coast of the United States, established primarily by the distance from Greenwich, England.

parent directory
The directory above "this" directory in the file system tree structure.

password
A character string that you type, usually just after your filename, to get access to the system. Keep your password secret, and change it when you think someone discovers what it is.

pathname
An identifier for the position of a file or directory with the tree structure of the file system. The three types of pathnames are: simple, absolute, and relative.

printer
A physical device that takes electronic signals, interprets them, and types them out onto paper.

printout
A piece of paper produced by a printer that has the image of the characters from a file on it.

program
A series of instructions that the system executes when you type the program name and commands associated with the program. A programmer writes the set of instructions and figures out how to get the system to use them properly.

"raw" file
For `nroff`, a file that has formatting commands embedded within it, so that the formatter can produce a formatted document from it.

read ahead
See **type ahead.**

redirecting output
Causing what the system types as the result of a command to go into a file, rather than onto the screen. Basic output redirection requires use of the > or >> symbols.

relative pathname
A series of directory names separated by /'s that locates a file or directory with respect to the current, or working directory. The other types of pathnames are: simple and absolute pathnames.

root directory
The "top" directory in the tree structure of the UNIX file system

rubout
To erase, or delete, off the screen.

save a file
See **write a file.**

screen
See **terminal screen.**

search a file
To look through the contents of a file, perhaps with `grep`, to find a certain character string.

simple pathname
A file or directory name, without mention of any associated directories, that one uses to access the file or directory.

special characters
One of a set of characters that have a meaning to the system, other than their meaning as a simple character. For instance, the exclamation point, `!`, has a special meaning for the history mechanism, and you shouldn't use it in a filename.

status line
For `vi`, the line the system types at the bottom of the interactive screen to provide information about the number of characters or lines in a file, or whether an instruction to write the file was successful.

string
A series of characters.

subdirectory
The child directory of a parent directory, or any directory below that child directory.

suspend
To halt, perhaps temporarily, the execution of a program.

system administrator
A person who tends the system, providing accounts, and hopefully solving any problems you have with the computer.

tablet
See **mouse tablet.**

terminal screen
A flat, rectangular part of the Sun workstation that you look at to see what you type to the system and what it types back.

text
Combinations of characters in strings, sometimes forming comprehensible language or command lines.

text editor
A program with which you can create files and modify them. For UNIX, the primary text editor is `vi`.

text formatter
A program with which you indicate more concisely arrangements and attributes of text, some of which are impossible to achieve without the formatter. The primary UNIX formatter is `nroff`.

tree structure
The way that many people describe the UNIX file system hierarchy, drawing an analogy between the structure of a tree and the structure of the file system. As with a tree, the file system originates with a root, a root directory, which "grows" child directories on its branches.

type ahead
The way that the system lets you type new commands while it is still interpreting and executing the current command.

universal time
See **Greenwich Mean Time.**

UNIX operating system
The operating system that runs on Sun workstations. See **operating system** for more information.

username
The character string with which you identify yourself to the system, usually assigned by your system administrator.

wild card characters
Characters that have a special meaning to the system, because they specify all filename character strings that have a certain attribute. For example, the wild card character asterisk, `*`, indicates a filename string of arbitrary length.

window
A portion of the screen in which you can type commands or execute programs while running the window system.

window system
A set of programs that allows you to divide up the screen into portions where you can type various commands and run many programs at the same time.

word
A character string, separated from other character strings, by spaces or tab characters.

working directory
The current directory in which, among other activities, you can type commands and list files. When you first log in, your working directory is your home directory.

work session
The time that you access the computer to work or play, between when you log in and when you log out.

workstation
A Sun system made up of a keyboard, a terminal screen, a mouse, and a

mouse tablet, which you can use for profitable work or healthy recreation.

write a file

For `vi`, to save the changes you have made to a file, so that when you next access the file the changes will still be there.

D

Bibliography

D

Bibliography

Look at the first appendix, called **Further Reading**, prior to perusing this list of bibliographic references.

General UNIX System Reference

Introducing the UNIX System, Henry McGilton and Rachel Morgan, McGraw-Hill Book Company, 1983. An introduction to UNIX for beginners and more sophisticated users. Covers the usual, plus communication facilities, editors, document formatting, software tool development, Berkeley UNIX, and system management. Packed with helpful examples.

The UNIX Programming Environment, Brian Kernighan and Rob Pike, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

The Bell System Technical Journal (BSTJ) Special Issue on UNIX, July/August, 1978. Contains papers describing developments and retrospective material.

The UNIX System, S. R. Bourne, Addison-Wesley Publishing Co., 1982. A comprehensive practical introduction for users from novices to experts.

UNIX Primer Plus, Mitchell Waite, Donald Martin, Stephen Prata, Howard W. Sams and Co., Inc., 1983. Examples for Berkeley and Bell Labs UNIX.

A User Guide to the UNIX System, Jean Yates and Rebecca Thomas, Osborne/McGraw-Hill, 1982. A tutorial for the 40 most used commands.

The UNIX Time-sharing System, D. M. Ritchie and K. L. Thompson, CACM, July 1974. An overview of the UNIX system for people interested in operating systems and worth reading by anyone who programs. Contains a remarkable number of one-sentence observations on how to do things right.

The Devil's DP Dictionary, Stan Kelly-Bootle. A humorous look at computer terminology.

Software Development Tools

Programming Tools for the Sun Workstation, Sun Microsystems Inc. Of general interest to anyone writing programs on the Sun system.

The C Programming Language, B. W. Kernighan and D. M. Ritchie, Prentice-Hall, 1978. Contains a tutorial introduction, complete discussions of all language features, and a reference manual.

Index

Revision History

Version	Date	Comments
A	17 February 1986	Beta 3.0 Release. Rework of **Getting Started** section of 2.0 Release of the *Beginner's Guide to the Sun Workstation*

Getting Started With UNIX: Quick Reference

This quick reference lists the commands presented in this manual concisely by function. Each listing includes a syntax diagram, and a brief description of the command.

1. Work Session

1.1. Log In

Type username to system login prompt.
Type password to password prompt.

1.2. Change Password

Type `passwd`, followed by old password, and repeat new password.

1.3. Log Out

Type `logout` or depending upon system setup.

2. File System

2.1. Create File

Type `cat >` *filename*, then text ending with or see **Editing Files.**

2.2. Make (or Create) Directory

Type `mkdir` *directory-name*.

2.3. Look at File

Type `cat` *filename*
or `more` *filename*.

2.4. Print File

Type `lpr` *filename*.

2.5. List Files and Directories

Type

`ls` for listing of current directory

`ls` *directory-name*
for listing of another directory

`ls` *filename*
for listing of a single file

`ls -t` or

`ls -t` *filename* or

`ls -t` *directory-name*
to get a listing reverse sorted by time of last modification

`ls -F |` or

`ls -F` *directory-name*
to get a listing that marks directory names by appending a / character to them.

2.6. Move (or Rename) Files and Directories

Type

`mv` *source-filename destination-filename*
to rename a file

`mv` *source-filename destination-directory*
to move a file into another directory

`mv` *source-directory-name destination-directory-name* to rename a directory, or move it into another directory.

2.7. Copy Files

Type

`cp` *source-filename destination-filename*
to copy a file into another filename

`cp` *source-filename destination-directory*
to copy a file into another directory

2.8. Remove (or Delete) File

Type

`rm` *filename*
to remove a file

`rmdir` *directory-name*
to remove an empty directory

`rm -r` *directory-name*
to remove a directory and its contents.

2.9. Change Working Directory

Type

`cd` to change directories to your home directory

`cd` *directory-name*
to change directories to another directory.

2.10. Find Name of Current Directory

Type `pwd`.

2.11. Pathnames

simple:	One filename or directory name to access local file or directory.
absolute:	List of directory names from root directory (first /) to desired filename or directory name, each name separated by /.
relative:	List of directory names from current position to desired filename or directory name, each name separated by /.

2.12. Directory Abbreviation

~	Home directory.
~*username*	Another user's home directory.
.	Working directory.
..	Parent of working directory.

3. Commands

3.1. Date and Time

Type `date`. For universal time (Greenwich Mean Time), type `date -u`.

3.2. Calendar

Type

`cal` *year*
for yearly calendar

`cal` *month-number year*
for monthly calendar

3.3. Wild Cards

`?` Single character wild card.

`*` Arbitrary number of characters.

3.4. Redirecting Output

System types output of command to file rather than screen, replacing current contents of file, if any. Type *command-name* > *filename*.

System types output of command to file rather than screen, appending to current contents of file, if any. Type *command-name* >> *filename*.

3.5. Basic Calculator

Type `bc` to enter interactive program. Type arithmetic expressions, using +, -, *, and / symbols, followed by To change number of decimal places, type `scale` = *number*.

4. Editing Files

Type `vi` to enter text editor, then any of following commands (in command mode, unless preceded by a `:`):

`a` to add text

`cc` to substitute a line with a string (enters insert mode)

`cw` to substitute, or change, a word with a string (enters insert mode)

`dd` to delete the entire line the cursor is on

`dw` to delete the word, or portion of word, under and after the cursor

`h` to move left, or "west," one character

`i` to insert text under the cursor (enters insert mode)

`j` to move down, or "south," one line

`k` to move up, or "north," one line

`l` to move right, or "east," one character

`o` to insert text on a new blank line after the current line (enters insert mode)

`O` to insert text on a new blank line before the current line (enters insert mode)

`s` to substitute a character with a string (enters insert mode)

`x` to delete the character under the cursor

`:q` to quit `vi`

`:q!` to quit `vi`, without writing changes

`:w` to save, or write a file

5. Formatting Files

Construct source file to run through `nroff` formatter, including any of the following commands:

`.LP` to *left-justify* a paragraph

`.IP` to created an *itemized* paragraph (like this one)

`.ce` to center text on the page

`.ul` to underline portions of text

`.sp` to create a blank line space

`.br` to force the end of a line, a line break

To format the source file, type `nroff -ms` *source-filename*. You will probably want to redirect the output of `nroff` into a *destination-filename*, so you can print it out afterward.

6. Search Files

Type

`grep` *search-string filename*
to type out lines containing the string in a specific file

`grep` *search-string filename(s)*
to type out lines containing the string in more than one file

`grep -v` *search-string filename(s)*
to type out lines that **don't** contain the string

7. Timesavers

7.1. Aliases

To "alias," or abbreviate a command string with an alias string, type `alias` *alias-string command-string*.

8. History: Command Repetition

`!!` Repeat the entire last command line at any point in the current command line.

`!$` Repeat the last word of the last command line at any point in the current command line.

9. Run Command in Background: Job Control

To run a command in the background, as opposed the the more command method of running commands in the foreground, type a `&` after the command line. Then, you can type more commands to the command prompt, or even run more commands in the background for simultaneous command execution.

10. Online Documentation

To see online Man Pages, type `man` *command-name*.

Notes

Notes

Notes

Notes

Corporate Headquarters
Sun Microsystems, Inc.
2250 Garcia Avenue
Mountain View, CA 94043
415 960-1300
TLX 287815

For U.S. Sales Office locations, call:
800 821-4643
In CA: 800 821-4642

European Headquarters
Sun Microsystems Europe, Inc.
Sun House
31-41 Pembroke Broadway
Camberley
Surrey GU15 3XD
England
0276 62111
TLX 859017

Australia: 61-2-436-4699
Canada: 416 477-6745
France: (1) 46 30 23 24
Germany: (089) 95094-0
Japan: (03) 221-7021
The Netherlands: 02155 24888
UK: 0276 62111

Europe, Middle East, and Africa, call European Headquarters:
0276 62111

Elsewhere in the world, call Corporate Headquarters:
415 960-1300
Intercontinental Sales