Commands to Use While Selecting a Newsgroup

Basic Commands

h	display help information
q	quit `trn`
x	quit `trn`, abandon all changes to your `.newsrc` file
v	show what version of `trn` you are using

Start Reading Articles

SPACE	perform the default command
+	invoke thread selector to read the current newsgroup
y	read the current newsgroup
=	same as y, but start by showing list of subjects

Controlling Newsgroups

n	go to next newsgroup with unread articles
p	go to previous newsgroup with unread articles
^	go to first newsgroup with unread articles
$	go to the end of the list of newsgroups
g *newsgroup*	go to specified newsgroup
/ *pattern*	scan forward for newsgroup name containing pattern
? *pattern*	scan backward for newsgroup name containing pattern
/	scan forward for previous pattern
?	scan backward for previous pattern
u	unsubscribe to the current newsgroup
l *pattern*	list unsubscribed newsgroups containing pattern
L	list current state of newsgroups in `.newsrc` file
c	mark all articles in newsgroup as read [catch up]
A	abandon all read/unread changes to this newsgroup

Entering a Unix Command

! *command*	execute the specified Unix command
!	pause `trn` and start a shell

Figure 13-3. *Commands to use while selecting a newsgroup*

FreeWheel, Ex. 1133, Part 2 of 3

Commands to Use While Selecting a Thread

In this list, *id* refers to an article id: **a...z, 0...9**. By default, the letters **h, k, m, n, p, q,** and **y** are omitted, as they are used as commands.

General Commands

`h`	display help information
SPACE	perform the default command

Changing Newsgroups

`q`	quit current newsgroup, move to next newsgroup
`N`	go to the next newsgroup with unread news
`P`	go to the previous newsgroup with unread news

Displaying the Selection List

`>`	go to next page
`<`	go to previous page
`^`	go to first page
`$`	go to last page
`L`	switch among: terse/short/long form of list

Selecting/Unselecting Which Threads to Read

`id`	select/unselect specified thread
`id-id`	select/unselect specified range of threads
`y`	select/unselect current thread
`.`	same as `y`
`k`	mark the current thread as completely read [kill]
`,`	same as `k`
`m`	unmark the current thread
`\`	same as `m`
`/ pattern`	select all threads whose subject contains pattern

Figure 13-4. *Commands to use while selecting a thread* (continued next page)

Changing Position Within the Thread List	
n	move down to the next thread
]	same as n
p	move up to the previous thread
[	same as p

Start Reading Articles	
RETURN	start reading (if nothing selected, current thread)
Z	start reading (if nothing selected, all threads)
TAB	same as Z
D	mark unselected articles on page read; start reading
X	mark all unselected articles as read, start reading

Figure 13-4. *Commands to use while selecting a thread (continued)*

Commands to Use While Selecting an Article

Basic Commands

h	display help information
q	quit this newsgroup
SPACE	perform the default command
n	go to next unread article
^N	go to next unread article on same subject
+	enter thread selection mode
k	mark as read all articles with same subject [kill]
u	unsubscribe to the newsgroup
c	mark all articles in newsgroup as read [catch up]

Re-displaying the Current Article

^R	re-display the current article
v	re-display the current article with header [verbose]
^X	decode the current article using rot-13
X	decode the current page using rot-13
b	go back one page
^L	re-display the current page

Moving to Another Article

-	re-display the last article that was displayed
N	forward: to next article, whether read or unread
$	forward: to end of the last article
p	backward: to previous article
^P	backward: to previous article with same subject
P	backward: to next article, read or unread
^	backward: to first unread article

Figure 13-5. *Commands to use while selecting an article* (continued next page)

Moving Within the Thread

`>`	go to the next selected thread
`<`	go to the previous selected thread
`]`	go to article's child within the thread
`[`	go to article's parent within the thread
`}`	go to the leaf of the tree
`{`	go to the root of the tree
`t`	display the entire article tree and all its subjects

Using Article Numbers

`=`	display list of all the unread articles
number	go to article with the specified number
`#`	display the number of the last article

Searching for an Article

`/` *pattern*	forward: look for subject containing pattern
`/`	forward: repeat search using previous pattern
`?` *pattern*	backward: look for subject containing pattern
`?`	backward: repeat search using previous pattern

Responding to an Article

`r`	send a mail message to author of article [reply]
`R`	same as `r`, include original message
`f`	start `Pnews` program to create a followup article
`F`	same as `f`, include original message

Saving an Article

`s` *file*	save entire article to specified file
`w` *file*	same as `s`, but do not save the header [write]

Entering a Shell Command

`!` *command*	execute the specified shell command
`!`	pause `trn` and start a shell

Figure 13-5. *Commands to use while selecting an article* (continued)

Commands to Use While Paging Through an Article

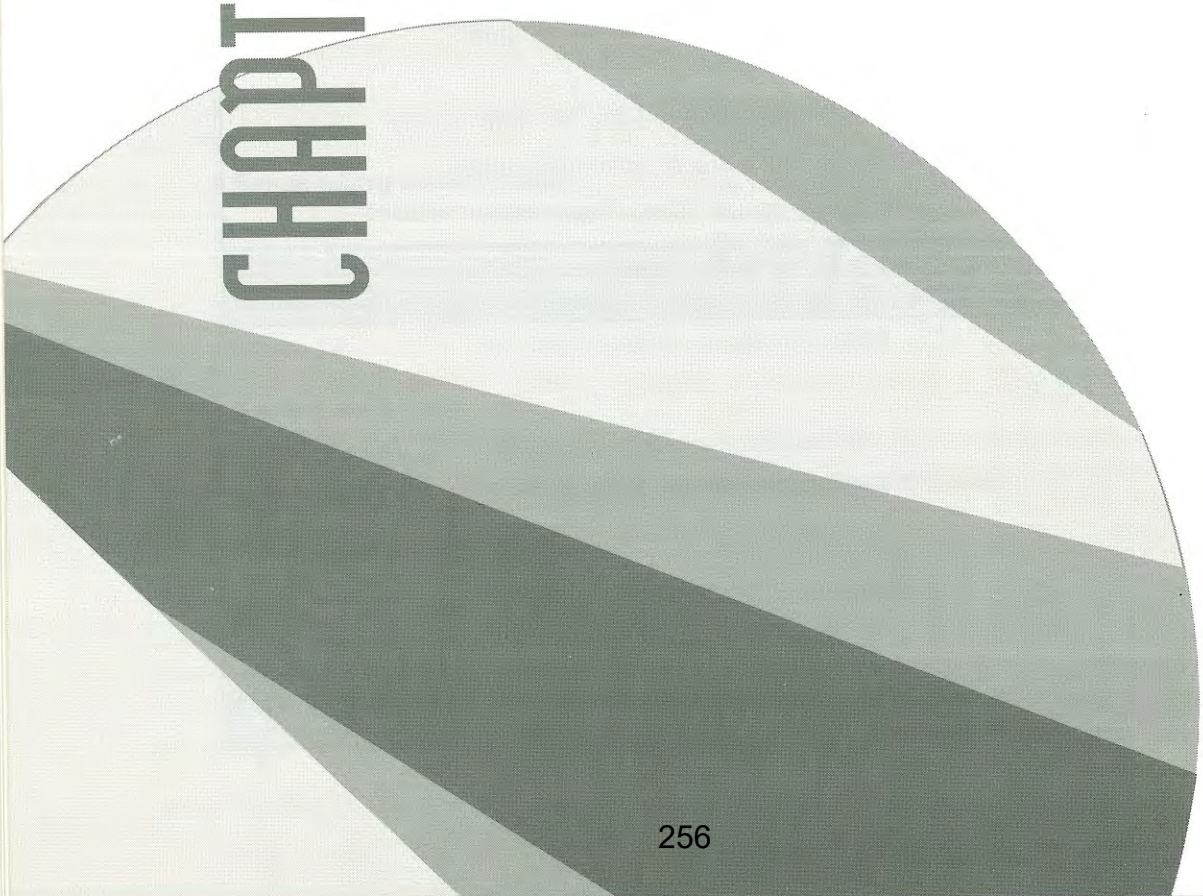
As you are paging through an article, you can use all the commands in Figure13-5 as well as the extra commands in this list.

Paging Through an Article

h	display help information
q	go to the end of the article
SPACE	display the next page
RETURN	display one more line
d	display half a page more
b	go back one page
x	decode the next page using rot-13
j	mark article as read and go to end [junk]
g *pattern*	forward: search for specified pattern
G	forward: repeat search using previous pattern
^G	while reading a digest: go to next subject
TAB	with followups: skip text from previous articles

Figure 13-6. *Commands to use while paging through an article*

CHAPTER 14



The nn Newsreader

We have talked a lot so far about Usenet, the worldwide system of over 5,000 discussion groups. In this chapter, we will explain how to read the news and post your own articles using the **nn** newsreader. Before you read this chapter, you should at least have skimmed the previous Usenet chapters. In particular, you should understand what Usenet is and how it works (Chapter 9), the conventions and terminology that we use (Chapter 10), and what is involved in using a newsreader (Chapter 11).

257

Overview of the nn Newsreader

The **nn** program was designed to solve the problem of too much news and too little time. **nn** makes it easy to choose which newsgroups you want to look at and, within those groups, which articles you want to read. **nn** is fast, easy to learn, and powerful. However, it is a complex program with many options and commands, and offers far more functionality than you will ever need.

The best strategy is to learn the basics and practice for a while. Once you have some experience, you can look at the command summaries at the end of this chapter and experiment with whatever looks interesting. Moreover, as you will see, **nn** makes it easy to display portions of the **nn** online manual from within the program itself. This means that, at all times, the official **nn** reference material is only a few keystrokes away.

To introduce you to **nn**, we will start with an overview of how it works. In the following sections, we will go into the details.

You start the program by entering the **nn** command. **nn** checks with your **.newsrc** file (see Chapter 11) and then displays the name of the first subscribed newsgroup that has unread news. You can choose to start with this group or you can select another one. Once you select a newsgroup, **nn** displays a list of unread articles. Each entry on the list is a short, one-line description of an article.

You mark only those articles that you want to read. After selecting all such articles, you type a command that tells **nn** to start displaying the articles.

As you read, there are a number of actions that you can take. You can save an article to a file, mail a copy of an article to somebody, kill all the articles with the same subject, decode the article using rot-13, and so on. You can also respond to an article, either by sending personal mail to the author, or by posting a followup article of your own.

In other words, at all times, you are either in one of two situations: selecting articles or reading articles. As you move from one newsgroup to another, you can change back and forth between selecting and reading.

At each point, there are a large number of commands available. As a reference, we have summarized all of the important **nn** commands in three lists at the end of the chapter.

Figure 14-2 shows the commands you can use at any time. Figure 14-3 shows extra commands that you can use while you are selecting articles. And Figure 14-4 shows extra commands that you can use as you are reading. You may want to take a minute now to glance at these summaries.

In the following sections, we will discuss the details of using **nn**. As you read, and as you practice, there are three things that we would like you to remember.

First, if you are typing a command that is waiting for input—and there are quite a few of them—you can cancel the command by pressing CTRL-G.

Second, for all single-letter commands, you do not have to press the RETURN or ENTER key after you type the command. All you have to do is press a single key and **nn** will react immediately. For example, to display help information, you press the **?** key. (When a Unix program reads input in this way, we say that it is operating in *cbreak mode*.)

Finally, whenever you are called upon to type a command, you can press SPACE to select the default choice. This choice will be the most reasonable thing to do in your current situation.

For example, when you are selecting articles, pressing SPACE will display the next page of the list. If you are at the end of the list, pressing SPACE tells **nn** to start reading. As you are reading, pressing SPACE repeatedly will page through an article, go to the next article, and ultimately (when there are no more articles to read) return you to the selection screen.

Moreover, when you enter a command that requires some sort of input, pressing SPACE will select the default choice, which is always a safe bet.

Hint

After selecting articles, you can read all the news you want just by using two keys. Use SPACE to page through your selected articles. If you get bored, press **k** (described later) to kill the rest of the articles in the thread.

Starting and Stopping

To start **nn**, you enter the **nn** command. There are several forms of this command and a large number of options that you can use. In this section, we will cover the most important of these choices.

First, you can enter the command by itself:

```
nn
```

This starts **nn** in a straightforward manner.

If you have never read the news before, **nn** will create a **.newsrc** file for you (see Chapter 11). Initially, your **.newsrc** file will contain all of the active newsgroups with each newsgroup subscribed. You will probably want to unsubscribe to most of the groups. There are two ways to do this. First, you can do it within **nn** by using the **U** command, although this will probably take a long

time. The more practical method is to use a text editor, such as **vi** or Emacs, and edit the **.newsrc** file for yourself (see Chapter 11).

Stopping **nn** is easy. When you are finished reading the last newsgroup to which you are subscribed, **nn** will quit automatically. Of course, since there is so much news, this may never actually happen. So, at any time, you can press **Q** to quit the program immediately.

There may be times when you are interested in reading only a single newsgroup. In that case, you can specify that group directly when you start **nn**. For example, say that you only want to read the **rec.humor** newsgroup. You can enter:



```
nn rec.humor
```

You can also specify more than one newsgroup:



```
nn rec.humor alt.tasteless.jokes
```

When you are finished with those newsgroups, **nn** will quit automatically.

If you would like **nn** to ask you what newsgroup to read, you can specify **-g** as part of the command:



```
nn -g
```

The **-g** is called an *option* or a *switch*. In this case, the **-g** option tells **nn** that you want it to ask you which newsgroup to "go to". You will see:



```
Enter Group or Folder (+./~)
```

Until you are an experienced user, don't worry about all the choices this message is offering. Simply type the name of the newsgroup you want and press RETURN. You will now see:



```
Number of articles (juasne)  (j)
```

Again, don't worry about the details. Just press SPACE and **nn** will start reading the newsgroup. After you are finished with that newsgroup, **nn** will quit automatically.

If you would like to have **nn** ask you for one newsgroup after another, you can use **-g** along with the **-r** (repeat) option:

```
nn -g -r
```

In general, when you enter a Unix command, you can combine options using a single - character, so in this case you can also use:

```
nn -gr
```

Using **-gr** is similar to **-g** alone except that **nn** will not quit after the first newsgroup. Rather, it will keep on asking you for one newsgroup name after another. When you are finished, press RETURN by itself (without specifying a newsgroup name) and **nn** will stop.

As we mentioned above, there are many options that you can use with the **nn** command. The only other ones we will discuss are a set of four options that can be combined in a particularly useful manner. These options are:

`-m`	merge all articles into an artificial "meta" group
`-x`	scan all articles: both read and unread
`-X`	look at all the newsgroups: subscribed and unsubscribed
`-s`	pattern show articles whose subject contains specified pattern

You can use these options together to tell **nn** to search through every article in every newsgroup looking for all the articles whose subject contains a specified pattern. The results will then be presented together as one large artificial newsgroup, called a *meta group*.

For example, say that you want to look at all the articles that contain the word "Unix" or "UNIX" in the subject. You would enter:

```
nn -mxX -sunix
```

(When you perform such a search, **nn** does not distinguish between upper- and lowercase letters. Thus "unix" will match both "Unix" and "UNIX".)

As an experiment, we used the previous command on a system that carries a large number of newsgroups. We found that **nn** took two minutes and 38 seconds to search 197,524 articles in 4,913 newsgroups. It found 849 articles that fit our specification.

To cut down on the number of newsgroups to search, you can specify newsgroup names. The special name **all** refers to all the newsgroups in a family. For example:

```
nn -mxX -sunix comp.all
```

This command searches all the newsgroups in the **comp** (computer) hierarchy.

Finally, there may be times when you do not want to actually read the news, but you just want to see if there are any new articles waiting for you. In such cases, you can use the **nncheck** command:

```
nncheck
```

If there are unread articles in the newsgroups to which you subscribe, **nncheck** will display a message like the following:

```
There are 1155 unread articles in 18 groups
```

Otherwise, you will see:

```
No News (is good news)
```

Be careful, though: this message may also mean that the news server is not working.

Hint

You can place an **nncheck** command in your initialization file so that each time you log in you will be told if there is news that you have not yet read. The name of your initialization file depends on which shell you are using. For the C-Shell, the file is **.login**; for the Bourne or Korn shell, it is **.profile**; and for the Bash shell, it is **.bash_profile**.

Displaying Help Information

One of the nice things about **nn** is that no matter what you are doing, there is plenty of help information available. First of all, when you are selecting or reading articles, you can press the **?** (question mark) key to display a summary of whatever commands are available.

We have summarized the most important of these commands in Figures 14-2, 14-3, and 14-4. Still, you may not always have this book with you—say, if you are

using **nn** on your portable computer while skydiving—and remembering the **?** command is a good idea.

nn also makes it easy to read the online manual without having to stop the program. You will remember that in Chapter 11 we explained that Unix comes with an online manual you can access by using the **man** command. For example, to display the manual entry for **nn**, you would normally use the Unix command:

```
man nn
```

Well, **nn** has a unique feature. While you are using **nn**, you can enter the command:

```
:man
```

nn will stop what it is doing and display a selection list for the entire **nn** manual. The sections of the manual will be presented using the same format that **nn** uses to present news articles. You can then use the exact same commands to select and read portions of the manual as you would with news articles.

Selecting Articles

In order to allow you to select the articles that you want to read, **nn** will show you a list. This list will display the authors and subjects of all the unread articles in the newsgroup, one page at a time.

Figure 14-1 shows a typical article selection list. **nn** can present this list in five different formats. If you would like to see the different formats, press the " (double quote) key repeatedly, and **nn** will change from one format to another.

The top line gives us some miscellaneous information. On the left, we see the name of the newsgroup. On the right, we see statistics about the articles: the number of unread articles in this newsgroup, the total number of unread articles in all the subscribed newsgroups, and the number of subscribed groups. In our example, we are reading the **misc.forsale** newsgroup. There are 35 unread articles in this group. Altogether, there are 817 unread articles in 14 newsgroups.

The main part of the list contains one line for each article. At the far left of each line is an *article id*, a single character that identifies the article. **nn** uses the letters of the alphabet, **a** through **z**, as article ids. If your screen holds more than 26 summary lines, **nn** will also use the digits **0** through **9**. In our example, there are 19 articles, marked **a** through **s**.

```
Newsgroup: misc.forsale                    Articles: 35 of 817/14 NEW

a*Reggie Jeeves      5  Works of Spinoza and Dostoevsky
b.Bertie Wooster    39  Banjolele, almost new
c.Catsmeat Potter-  33  >> reports signed by Rev. Aubrey Upjohn
d.Tuppy Glossop     29  Football boots
e Harold Pinker     11  >
f Dahlia Travers     4  Terra-cotta figure: Infant Samuel at Prayer
g.Bingo Little       8  Theatre poster of 'What Ho, Twing!!'
h Stilton Cheesewr  10  Rowing trophy from a bump supper
i Tom Travers       16  Full set of back copies: Milady's Boudoir
j Oliver Sipperly   15  Policeman's helmet from Coventry Street
k*Roderick Glossop  30  Dead fish and 23 cats
l*Dog-Face Rainsby  12  >
m*Claude Wooster    28  >
n*Eustace Wooster   13  >
o R.E.R. Psmith     19  Umbrella
p Watkyn Bassett    14  -
q Mrs. Bingo        53  Full set of Rosie M. Banks novels
r Gussie Fink-Nott  31  Picture of Madeline Bassett; Newt food
s T. L. Nipper      17  Dish with separate water compartment

-- 15:49--SELECT--help:? -----Top 5%-----
```

Figure 14-1. *A typical article selection list*

To the right of the article id are the name of the author, the number of lines in the article, and the subject. In the subject column, a > character indicates a followup. This allows you to identify threads. In our example, articles **d** and **e** form a short two-article thread. Articles **k** though **n** form a longer four-article thread. If an article is a followup to a followup, you will see two > characters (>>). This is the case for article **c**.

On occasion, more than one unrelated article may have the same subject. In such cases, **nn** will display the subject for the first article and a - (hyphen) character for the others. In our example, we see this with articles **o** and **p**.

On the bottom line, **nn** displays a few items of information. At the far left is the time. In our example, it is 3:49 PM (15:49). Next is a reminder that we are looking at the article selection list, followed by another reminder, telling us that we can display a help summary by pressing the **?** key. To the right, **nn** tells us where we are in the full list. In our example, we are at the top of the list, and the last article shown is 5% of the way through the list.

Selecting an article to read is easy. All you have to do is type the article id. For example, to select article **a**, just press the **a** key. If you want to select a range of articles, type the two article ids, separated by a - (hyphen). For example, to select articles **k** through **n**, type **k-n**.

To select an entire thread, type the article id that marks the beginning of the thread, followed by an * (asterisk). For example, to select all the articles that have the same subject as article **k**, type **k***.

When you select an article, **nn** will highlight it on your screen. If your terminal or computer cannot support highlighting, **nn** will mark selected articles with an * character after the article id. In our example, you can see that we have selected articles **a**, **k**, **l**, **m**, series and **n**.

To unselect an article, simply type the article id of the article once again. For example, once article **a** is selected, you can unselect it by pressing the **a** key a second time. You can also unselect by typing a range of article ids (**k-n**) or a thread specification (**k***).

At any time, your cursor will be on one of the articles in the list. This is the current article. You can move up and down the list by using the UP and DOWN keys (the keys with the arrows). However, you can also use the , (comma) and / (slash) keys, which is a lot more handy. The , moves down and the / moves up. The . (period) key selects the current article. Making your selections in this way is convenient because, on many keyboards, these three keys are next to one another.

Hint

Put the middle fingers of one hand on the , (comma), . (period), and / (slash) keys. Next, place your thumb on SPACE. You can now zip through a list of articles with the speed of a greased jackrabbit.

Once you have selected all the articles you want, there are several ways to tell **nn** that you want to start reading. Remember that pressing SPACE moves to the next page of articles. If you are already on the last page, pressing SPACE will start reading.

Alternatively, you can press **Z** or **X** at any time. Both of these commands tell **nn** to start reading immediately. The only difference is that, after you are finished reading, **Z** returns you to the same newsgroup, while **X** goes on to the next newsgroup.

There are two basic strategies for selecting and reading. Some people like to skim through the selection pages, make all their selections, and then read the articles. Other people like to select and read, one page at a time.

Moving to Another Newsgroup

As you are selecting articles, there are several commands that you can use to move to another newsgroup. To move to the next subscribed newsgroup in the sequence, press **N** (next).

There is also a **P** (previous) command, but it does not move backwards in the sequence of newsgroups. Rather, **P** moves to the last newsgroup that you looked at. This means that if you press **P** twice, you get back to where you started.

There is also a convenient way to move back and forward: press **A** (ahead) or **B** (back). These commands will move throughout the entire newsgroup list, to both subscribed and unsubscribed newsgroups.

When you use either of these commands, you will see the name of a newsgroup followed by a set of possible commands. For example:

```
Enter alt.fan.wodehouse (UNSUB) ?  (ABGNPy)
```

In this case, we are being asked if we want to enter the newsgroup named **alt.fan.wodehouse**, which is currently unsubscribed.

At this point, you can press SPACE or **y** (yes) to move to this group. You can also press **A** to move ahead, **B** to move back, **G** to go to a specific group (see below), **N** to move to the next subscribed group with unread news, or **P** to move to the previous subscribed group with unread news.

Hint

You can quickly skim through the full list of newsgroups by pressing **A** or **B** repeatedly.

Once you find a newsgroup you like, press SPACE. You will see:

```
Number of articles (juasne)  (j)
```

Until you become an experienced **nn** user, don't worry about what this means. Just press SPACE once again and you will jump right to the newsgroup you want.

The final way to change newsgroups is to jump directly to a specific newsgroup. To do so, press **G** (go to). After you do, you will see:

```
Group or Folder (+./~ %=sneN)
```

Once again, until you become experienced, ignore the options. Just type the name of the newsgroup you want and press RETURN. You will then see:

```
Number of articles (juasne)  (j)
```

Press SPACE to jump to the group.

Hint

The Fundamental Rule of **nn**:

> Whenever you are unsure what to do, press SPACE. (If only life were that simple.)

Reading Articles

While you are reading an article, there are many commands that you can use. The basic commands are summarized in Figures 14-2 and 14-4. For the most part, these commands are straightforward and, with a little practice, you will have no trouble at all. To get you started, we will introduce you to the most important ones.

As you no doubt expect, the single most important command is SPACE. Pressing SPACE repeatedly will allow you to read an entire article, from one page to the next. When the article is finished, pressing SPACE once again will move to the next article. At the end of the last article, pressing SPACE will move you back to the selection screen.

As you read, you may decide that a particular thread is boring. If so, press **k** to kill all the remaining articles in that thread. To go directly to the next article, press **n**; to go to the previous article, press **p**.

If you are not sure that you want to read an entire thread, you can mark only the first article while you are selecting. Later, when you actually read the article, you can choose to read the rest of the thread by pressing * (asterisk).

If you happen upon a potentially offensive article that was encoded using rot-13 (see Chapter 11), you can display it in decoded form by pressing **D**.

When you encounter an article that is especially interesting or useful, there are two ways that you can preserve it. First, you can save the article in a file by using the **s**, **o**, or **w** command. The **s** command saves the full header along with the article. The **o** command saves an abbreviated header along with the article. And the **w** (write) command saves the article only, with no header.

The second way to preserve an article is to use the **m** command to mail a copy of the article to someone (including yourself).

Finally, there are two ways to respond to an article. First, you can send a private mail message to the author by using the **r** command. You will be asked:

```
Include original article?
```

Press either **y** (yes) or **n** (no). In either case, **nn** will start your text editor for you, to allow you to compose your reply. If you chose yes, **nn** will include the text of the article when it starts the editor, so you can include parts of it in your reply. Once you quit the text editor, **nn** will resume control and you will see the following:

```
a)bort e)dit h)old m)ail r)eedit s)end v)iew w)rite
Action: (send letter)
```

You now have several straightforward choices. To make a choice, you must type the specific letter and press RETURN. For example, if you decide not to send your reply after all, type **a** and then press RETURN. If you press RETURN by itself, **nn** will send the letter by default, so you don't really have to use the **s** command.

The second way to respond to an article is to post a followup article of your own. We will discuss this in the next section.

Posting an Article

There are three ways to post an article with **nn**. No matter how you do it, posting an article is easy. **nn** will lead you through the process; all you have to do is answer the questions. When **nn** is ready for you to compose the article, it will automatically start your text editor. When you quit the editor, **nn** will regain control.

The first way to post an article is as a followup while you are reading the news. If you are at the article selection screen, just move to the article to which

you want to respond and press **F**. If you are currently reading an article, you can press either **f** or **F**.

In either case, **nn** will ask if you want to include a copy of the original article in your followup. The default is yes, so if you want to include the original, you can press either **y** or SPACE; otherwise, press **n** (no). If you answer **y** (yes), the original article will be inserted with a > character at the beginning of each line.

Hint

It is considered polite to delete all the unnecessary lines from a followup.

The second way to post an article while you are using **nn** is by entering the **:post** command at any time. **nn** will ask you for all the relevant information: newsgroup, subject, and so on. You can use this command to post an article to any newsgroup using any subject you want.

Finally, when you are not using **nn**, you can post an article by entering the **nnpost** command. The easiest way to use this command is to enter it without any options:

```
nnpost
```

The program will ask you everything it needs to know, just as if you had entered **:post** within **nn**.

If you would like to learn more about the **nnpost** command, you can read about it in the online manual. You will see that you can use options to specify some of the information that the program needs. In addition, you will find out how to set up automated postings.

The nn Family of Commands

The **nn** program is actually one of a family of commands. The entire family is shown in Table 14-1.

We have already discussed the **nn**, **nncheck**, and **nnpost** commands. We won't cover the other commands here. However, once you get some experience, we recommend that you take a few moments and check out these commands in

Command	Purpose
`nn`	read the news
`nncheck`	check for unread articles in subscribed newsgroups
`nngoback`	mark specified days' articles as unread
`nngrab`	find all articles whose subject contains a pattern
`nngrep`	display all newsgroup names that contain a pattern
`nnpost`	post news articles
`nntidy`	clean and adjust your `.newsrc` file
`nnusage`	display statistics showing who has been using `nn`

Table 14-1. *The* **nn** *Family of Commands*

the online manual. (You can use either the Unix **man** command, or the **:man** command from within **nn**.)

In particular, **nngrab** is a fast alternative to the cryptic combination of options that we used earlier in the chapter (**nn -mxX -s**) to search all the newsgroups for articles whose subject contains a specific pattern.

Commands That Are Always Available

Basic Commands

SPACE	do the most reasonable thing/accept the default
Q	quit nn
CTRL-G	cancel a command that is waiting for input
`:post`	post a new article

Displaying Help

?	display summary of available commands
`:help`	show extra sources of help
`:man`	read the nn entries in the online manual

Controlling Newsgroups

G	go to a newsgroup out of sequence
U	unsubscribe/subscribe to the current newsgroup
Y	show overview of newsgroups with unread articles

Entering Unix Commands

! *command*	execute the specified Unix command
!	pause nn and start a new shell

Figure 14-2. *Commands that are always available*

Commands to Use While Selecting Articles

In this list, *id* refers to an article id: **a...z, 0...9**

Selecting/Unselecting Which Articles to Read

id	select/unselect specified article
id-id	select/unselect specified range of articles
*id**	select/unselect articles with same subject as *id*
,	move down one line
.	select/unselect current article, then move down
/	move up one line
*	select/unselect same subject as current article
@	reverse selections on current page
=. RETURN	select all articles in the newsgroup
~~	unselect all articles in the newsgroup

Displaying the Selection List

SPACE	go to next page, unselected articles become read
>	go to next page
<	go to previous page
^	go to first page
$	go to last page
"	change the layout

Start Reading Articles

SPACE	start reading (when on last page)
Z	start reading; return to newsgroup when done
X	start reading; go to next newsgroup when done

Moving to Another Newsgroup

N	move to next subscribed newsgroup in the sequence
P	move to last newsgroup that you looked at [previous]
B	move back one newsgroup in the sequence
A	move ahead one newsgroup in the sequence
G	go to a specified newsgroup

Figure 14-3. *Commands to use while selecting articles* (continued next page)

Saving an Article

S	save full header plus article to a file
O	save abbreviated header plus article to a file
W	save article only (no header) to a file [write]

Responding to an Article

R	send mail message to author of article [reply]
F	create a followup article
M	mail a copy of the article to someone

Figure 14-3. *Commands to use while selecting articles* (continued)

Commands to Use While Reading Articles

Moving Within an Article

SPACE	forward: one page
DOWN	Same as SPACE
d	forward: one half page [down]
RETURN	forward: one line
$	forward: to last page
BACKSPACE	backward: one page
UP	same as BACKSPACE
u	backward: one half page [up]
^	backward: to first page
g*line*	go to specified line
/*pattern*	search forward for specified pattern
.	repeat previous search
TAB	followup articles: skip text from older article

Displaying an Article

h	re-display the current article with full header
D	decode the current article using rot-13
c	omit multiple spaces and tabs [compress]

Move to Another Article

SPACE	go to next article (if on last page)
n	go to the next selected article
p	go to the previous selected article
k	mark as read articles with same subject [kill]
*	go to next article with same subject

Return to Selecting Articles

SPACE	return to selecting (if last page of last article)
=	return to selecting current newsgroup
N	go to next newsgroup
X	go to next newsgroup, mark all articles as read

Figure 14-4. *Commands to use while reading articles* (continued next page)

Saving the Current Article

`S`	save full header plus article to a file
`s`	same as `S`
`O`	save abbreviated header plus article to a file
`o`	same as `O`
`W`	save article only (no header) to a file [write]
`w`	same as `W`

Responding to the Current Article

`R`	send mail message to author of article [reply]
`r`	same as `R`
`F`	create a followup article
`f`	same as `F`
`M`	mail a copy of the article to someone
`m`	same as `M`

Figure 14-4. *Commands to use while reading articles* (continued)

CHAPTER 15

The tin Newsreader

We have talked a lot so far about Usenet, the worldwide system of over 5,000 discussion groups. In this chapter, we will explain how to read the news and post your own articles using the **tin** newsreader. Before you read this chapter, you should at least have skimmed the previous Usenet chapters. In particular, you should understand what Usenet is and how it works (Chapter 9), the conventions and terminology that we use (Chapter 10), and what is involved in using a newsreader (Chapter 11).

277

Overview of the tin Newsreader

The **tin** program was designed to let you skim through a large number of newsgroups and a large number of articles. **tin** makes it easy to choose which newsgroups you want to look at and, within those groups, which threads of articles you want to read. **tin** is fast, easy to learn, and powerful. However, it is a complex program with many options and commands, and offers far more functionality than you will ever need.

The best strategy is to learn the basics and practice for a while. Once you have some experience, you can look at the command summaries at the end of this chapter and experiment with whatever looks interesting. When you get really good, you can read the reference material on **tin** in the online manual and find all kinds of cool stuff.

To introduce you to **tin**, we will start with an overview of how it works. In the following sections, we will go into the details.

You start the program by entering the **tin** command. The first thing **tin** does is read your **.newsrc** file (see Chapter 11) and compare it against the master list of active newsgroups. If there are new newsgroups, **tin** asks you whether or not you want to subscribe.

Once the check for new groups is finished, **tin** presents you with a newsgroup selection list. This is a list of newsgroups from which you can choose what you want to read.

After you select a newsgroup, **tin** shows you a thread selection list in which all the different threads for this newsgroup are shown. You can now choose a thread to read.

As you read, there are a number of actions that you can take. You can save an article to a file, mail a copy of an article to somebody, kill all the articles with the same subject, decode the article using rot-13, and so on. You can also respond to an article, either by sending personal mail to the author, or by posting a followup article of your own.

In other words, at all times, you are in one of three situations: selecting a newsgroup, selecting a thread, or reading articles. (Actually, **tin** has two other environments: spool directories and newsgroup indexing. However, these topics are not important to most people and we won't discuss them in this chapter.)

At each point, there are a large number of commands available. As a reference, we have summarized all of the important **tin** commands in four lists at the end of the chapter.

Figure 15-4 shows the commands you can use at any time. Figure 15-5 shows commands that you can use while you are selecting a newsgroup. Figure 15-6 shows commands you can use while you are selecting a thread to read. And Figure 15-7 shows commands that you can use as you are reading. You may want to take a minute now to glance at these summaries.

Within the summaries, we have followed the common Unix convention of indicating CTRL keys by using the ^ character. For example, when you see ^D, it means CTRL-D. (That is, hold down the CTRL key and press D.)

In the following sections, we will discuss the details of using **tin**. As you read, and as you practice, there are three things that we would like you to remember.

First, in almost all circumstances, you can press **h** to display a help summary of all the commands that are currently available.

Second, for all single-letter commands, you do not have to press the RETURN or ENTER key. All you have to do is press a single key and **tin** will react immediately. For example, to display help information, you press the **h** key. (When a Unix program reads input in this way, we say that it is operating in *cbreak mode*.)

Finally, there will be times when **tin** will ask you a question and present you with a list of possible answers. In such cases, **tin** will suggest one of the answers as being the most likely response. This is the default. If you would like to accept this suggestion, you can simply press RETURN.

For example, as you are reading an article, you can save a copy of the article to a file by using the **s** command. When you do, you will see the following:

```
Save a)rticle, t)hread, h)ot, p)attern, T)agged articles, q)uit: a
```

tin is offering you five choices. All you have to do is press the key that you want. Notice that after the colon, **tin** has put a default of **a**. Since this is the default, pressing RETURN is the same as pressing **a**.

Hint

If your terminal or computer has cursor control keys—the keys with the arrows: UP, DOWN, LEFT, and RIGHT—**tin** makes great use of them.

In general, the UP and DOWN keys move up or down one line (or, where appropriate, one page).

The LEFT and RIGHT keys move up or down a logical level. For example, when you are selecting a newsgroup, RIGHT moves you to the thread selection list. When you are selecting a thread, LEFT moves you back to the newsgroup selection list.

Once you get a little practice, this will all make perfect sense. In the words of Iain Lea (Mr. **tin**), "You can drive **tin** all day long just by using these four keys."

Customizing Your Working Environment

tin has more ways to change settings and to customize your environment than you would believe. Unfortunately, modifying most of these settings requires a degree of knowledge that is well beyond that of the beginning or casual user.

However, **tin** does make it easy for you to change some of the more basic settings. At any time, you can press the M (menu) key. (Try it.) This will bring up a list of a limited number of settings. Each of the settings will have a number. To change a setting, type the number, press RETURN, and follow the instructions.

After you select a setting to change, you may see the following instruction:

```
<SPACE> toggles & <CR> sets.
```

This means that the setting you are modifying has a fixed number of choices. Pressing SPACE will change from one choice to the next. When you are finished, and you want to set your final choice, press RETURN ("Carriage Return").

Starting and Stopping

To start **tin**, you enter the **tin** command:

```
tin
```

If you have never used **tin** before, you will see a screen of introductory information. This screen only appears the first time you use **tin**. Read this over, but don't worry if you don't understand it all. Everything you need to know to get started is in this chapter.

When **tin** starts, it goes through a number of initialization procedures. You will see messages like:

```
Reading news active file...
Reading attributes file...
Reading newsgroups file...
```

When you see the last line, it means that **tin** is reading your **.newsrc** file and comparing it against the news server's master list of active newsgroups. If there are any new newsgroups, **tin** will add them to your **.newsrc** and ask you if you want to subscribe.

If you have never read the news before, **tin** will create a **.newsrc** file for you (see Chapter 11). Initially, your **.newsrc** file will contain all of the active newsgroups with each newsgroup subscribed. You will probably want to unsubscribe to most of the groups. There are two ways to do this. First, you can do it within **tin** by using the **U** or **u** command, although this will probably take a long time. The more practical method is to use a text editor, such as **vi** or Emacs, and edit the **.newsrc** file for yourself (see Chapter 11).

To stop **tin**, you can use the **Q** (quit) command at any time. Simply press the **Q** key. Notice that this is an uppercase (capital) letter. The **q** command is used to leave whatever part of the program you happen to be using and return to the previous level. The **Q** command will quit **tin** outright.

Most of the time, there is a better way to start **tin** than the simple command we mentioned above. You will remember that we said that the first thing that **tin** does is to check to see if there are new newsgroups that are not listed in your **.newsrc** file. Although this is a good idea, it can be time consuming, perhaps taking even several minutes each time.

Most people prefer to have **tin** skip this check. You can do so by using the command:

```
tin -q
```

The **-q** is called an *option* or a *switch*. In this case, the **-q** option tells **tin** to perform a "quick" start. Using **-q** will start **tin** faster, but will not, of course, pick up any new newsgroups. For this reason, we recommend starting **tin**, say, once every week or two, without the **-q** option, just to see what is new.

There may be times when you are interested in reading only a single newsgroup. In that case, you can specify that group directly. For example, say that you only want to read the **rec.humor** newsgroup. You can enter:

```
tin rec.humor
```

You can also specify more than one newsgroup:

```
tin rec.humor alt.tasteless.jokes
```

When you start **tin** in this way, it will show you only the newsgroups you specify.

tin has quite a few options that you can use. Most of the time, you will not need any of them except for **-q**. However, we will discuss a few of the more useful ones in case you do want to try them. For more information about options, you can read the entry for **tin** in the online manual by entering the Unix command:

```
man tin
```

If you would like to display a quick list of all the possible options, you can use the **-h** (help) option:

```
tin -h
```

Remember that we said that the first time you use **tin** it will display a screen of introductory information. If you would like to display this screen again sometime, you can do so by using the **-H** (help summary) option:

```
tin -H
```

Another option that you may want to use is **-w** (write). You use this when you want to start **tin** only for the purpose of posting a news article. We will discuss this later in the chapter.

The last option we will mention is **-z**:

```
tin -z
```

This tells **tin** to start up only if there are articles that you have not yet read. You can combine this option with the names of one or more newsgroups. This is handy when you want to check quickly if there is anything of interest to you.

For example, say that you follow the **rec.humor** newsgroup religiously. (That is, you start each day by covering your head and reading jokes.) You can use the following command to start **tin** only if there are new articles in that newsgroup:

```
tin -z rec.humor
```

Hint

You can place a **tin -z** command in your initialization file so that each time you log in you will automatically start **tin** if there is news that you have not yet read. The name of your initialization file depends on which shell you are using. For the C-Shell, the file is **.login**; for the Bourne or Korn shell, it is **.profile**; and for the Bash shell, it is **.bash_profile**.

The Newsgroup Selection List

To read the news, the first thing you must do is choose which newsgroup you want to look at. To help you to make a choice, **tin** will display a list of newsgroups, one page at a time. Figure 15-1 shows a typical newsgroup selection list.

The top line of Figure 15-1 tells us that we are selecting newsgroups. The number in parentheses is the number of subscribed newsgroups, in this case, 157. In the top right-hand corner, we see a reminder that we can display a help summary by pressing the **h** key.

Next, follow as many newsgroups as will fit on one page. In our example, we see 16 newsgroups at a time. The entry for each newsgroup shows us an identification number, the number of unread articles, the name of the newsgroup, and a short description. In our example, three of the newsgroups (4, 5, and 14) have no unread articles.

The descriptions to the right are taken from a master file that is distributed regularly to news servers. You will find that not all the descriptions are informative and that many of them are missing. If you decide that you would rather not see the descriptions, you can tell **tin** to omit them by pressing the **d** (description) key. To bring back the descriptions, press **d** again. If you want to change this setting permanently, you can use the **M** (menu) command to modify the **Show description** setting.

The bottom part of the group selection list is a short summary to help you with the most important commands. After we discuss the commands (in the next section), this summary will make sense. If you decide that you would like to get rid of this summary—and make more room for newsgroup names—press **H**. To bring back the summary, press **H** again.

You will notice that our example shows a few newsgroups that have no unread articles. Quite possibly, you will not want to look at such newsgroups as there is nothing to read. If so, you can press the **r** key. This tells **tin** to display only those newsgroups that have unread articles. To change back, press **r** once

```
                        Group Selection (157)

1  3   alt.1d                        One-dimensional imaging, and the think i
2  7   alt.3d                        Three-dimensional imaging
3  27  alt.abortion.inequity         The inequity of abortion
4      alt.abuse.offender.recovery   Helping offenders recover
5      alt.abuse.recovery            Helping victims of abuse recover (Moder
6  10  alt.activism                  Activities for activists
7  3   alt.activism.d                A place to discuss issues in alt.activi
8  27  alt.adoption                  For those involved with or contemplatin
9  5   alt.aeffle.und.pferdle        German TV cartoon characters
10 17  alt.agriculture.fruit         Fruit farming and agriculture
11 143 alt.agriculture.misc          Agriculture and farming
12 27  alt.aldus.freehand            Aldus Freehand software
13 54  alt.aldus.misc                Other Aldus software products
14     alt.aldus.pagemaker           All about Aldus PageMaker
15 36  alt.alien.visitors            Space aliens on Earth! Abduction! Gov't
16 131 alt.amateur-comp              The amateur computerist

    <n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unread,
   s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out
```

Figure 15-1. *A typical group selection list*

again. If you want to change this setting permanently, you can use the **M** (menu) command to modify the **Show only unread** setting.

Selecting a Newsgroup

To start reading a newsgroup, all you have to do is move to the group you want and press RETURN. While you are working with the newsgroup selection list, there are many commands that you can use. As well as the general commands, there are commands for moving within the newsgroup list, and for telling **tin** to start reading when you have selected a newsgroup.

The general commands are summarized in Figure 15-4. The newsgroup selection commands are summarized in Figure 15-5. In this section, we will discuss the most basic of these commands. When you get some time, check out Figure 15-5 and try some of the other commands.

tin displays only as many newsgroup names as can fit on a single page (that is, one screenful). To display the next page, press PAGE DOWN. To move back one page, press PAGE UP. If your terminal or computer does not have these keys (or if you prefer not to use them), you can use SPACE (PAGE DOWN) and **b** (PAGE UP) instead.

At all times, one of the newsgroup names will be highlighted. To move down or up one line, use the DOWN and UP keys. Again, if you do not have these keys (or you do not want to use them), there are alternatives. You can use **j** (down) and **k** (up) instead.

To jump directly to a particular newsgroup, you have two choices. First, you can enter its number. For example, to jump to newsgroup #4, press **4** and then press RETURN.

Second, you can press **g** (go to), type the name of the newsgroup, and press RETURN. When you use the **g** command, **tin** will suggest the last newsgroup you jumped to as a default choice. If you want to jump to this newsgroup again, just press RETURN. (It will make sense when you see it.)

Once you have chosen the newsgroup you want, you can enter it by pressing RIGHT or RETURN. Alternatively, you can press TAB to automatically enter the next group that has unread articles.

Finally, when you are finished reading the news, press **q**, **Q**, or LEFT to quit **tin**.

The Thread Selection List

Once you have selected a newsgroup, **tin** shows you a list of threads. You examine this list and select the first thread that you want to read. Figure 15-2 shows a typical thread selection list.

The top line shown in Figure 15-2 tells us the name of the newsgroup, in this case **soc.culture.british**. Directly after the name, **tin** shows us the number of threads, the number of articles, the number of killed (junked) articles, and the number of hot articles. (Hot articles are those that are selected automatically according to a predefined criterion. For more details, see the reference manual.) In our example, there are 69 threads of 245 articles. None of these articles are killed or hot.

Next, follow as many thread summaries as will fit on one page. In our example, we see 15 threads. The summary of each thread shows us an

```
                soc.culture.british (69T 245A 0K 0H)        h=help

 1  + 2  Empress of Blandings' silver medal    Clarence Emsworth
 2       A world full of loonies               Alaric Dunstable
 3  + 3  Making a proper marriage              Constance Keeble
 4  + 13 Five ways to call a pig               George Wellbeloved
 5  + 6  The best dog biscuits to use          Freddie Threepwood
 6    3  Choosing port                         Sebastian Beach
 7       Efficiency                            Rupert Baxter
 8  +    Things I don't understand             Veronica Wedge
 9  +    The Queen of Matchingham              Gregory Parsloe
10  + 1  Poetry for Polly Pott                 Ricky Gilpin
11  +    My life with Sue Brown                Ronald Fish
12  + 63 Does the Ickenham System work?        Galahad Threepwood
13  +    On The Care Of The Pig                Augustus Whipple
14  +    Housekeeping and etiquette            Mrs. Twemlow
15    16 Canadian poetry                       Ralston McTodd
```

Figure 15-2. *A typical thread selection list*

identification number, the number of followup articles, and the subject and author of the first article. For example, thread #12 contains the original article plus 63 followups. The thread was started by Galahad Threepwood, who has initiated a discussion as to whether or not the Ickenham System actually works. Thread #2 contains only a single article with no followups.

You will notice that, after the thread number, some of the summaries have a + (plus) character. This indicates that the thread has not yet been read. In our example, threads 2, 6, 7, and 15 have been read.

As you can see, the thread selector list shows both the subject and the author of the first article in the thread. If the subject descriptions are long, they will be truncated. To make more room, you can press **d** to tell **tin** to display only the subject. To change back and display both the subject and author, press **d** again. If you want to change this setting permanently, you can use the **M** (menu) command to modify the **Show author** setting.

Similarly, you can tell **tin** to display only the unread threads by pressing **r**. To change back and display both read and unread threads, press **r** again. If you want to change this setting permanently, you can use the **M** command to modify the **Show only unread** setting.

Selecting a Thread to Read

Once you have selected a newsgroup, **tin** shows you a list of threads. To read the articles, all you have to do is move to a thread and press RETURN.

While you are working with the thread selection list, there are many commands that you can use. As well as the general commands, there are commands for moving within the thread list, and for telling **tin** to start reading when you have selected a thread.

The general commands are summarized in Figure 15-4. The thread selection commands are summarized in Figure 15-6. In this section, we will discuss the most basic of these commands. When you get some time, check out Figure 15-6 and try some of the other commands.

tin displays only as many thread descriptions as can fit on a single page (that is, one screenful). To move throughout the list, you can use the same commands as with the newsgroup list.

To display the next page, press PAGE DOWN or SPACE. To move back one page, press PAGE UP or **b**. To move down one line, press DOWN or **j**. To move up one line, press UP or **k**.

To jump directly to a particular line, type the number followed by RETURN. For example, to jump to thread #4, press **4** and then press RETURN.

To go to the next unread thread, press **N**; to go the previous unread thread, press **P**. If you press **K** (kill), it will mark a thread as read and move to the next line.

Once you have chosen the thread you want, you can start reading by pressing RIGHT or RETURN. Alternatively, you can press TAB to automatically start reading the next unread thread.

If you want to preserve a thread, you have three choices. You can press **m** to mail a copy to someone (including yourself), **o** to print it, or **s** to save it to a file.

Finally, when you are finished reading, you can return to the newsgroup selection list by pressing **q** or LEFT. To quit **tin** completely, press **Q**.

Reading Articles

Figure 15-3 shows a typical article as displayed by **tin**. Across the top three lines, we see descriptive information about the article. **tin** automatically extracts this information from the header.

The top line in Figure 15-3 shows the time and date that the article was posted, the name of the newsgroup, and information about the thread. The

second line shows the article number, the subject, and the number of responses (followups). The third line shows the author's mail address and description.

Next comes the body of the article, including, in this case, a signature.

Finally, the last three lines contain a short summary to help you with the most important commands. If you decide that you would like to get rid of this summary—and make more room for the article—press **H**. To bring back the summary, press **H** again.

As you are reading an article, there are many commands that you can use. As well as the general commands, there are commands for paging through the article, moving to a different article, preserving the article, and responding to the article.

The general commands are summarized in Figure 15-4. The other commands are summarized in Figure 15-7. In this section, we will discuss the most basic of these commands. When you get some time, take a look at Figure 15-7 and experiment for yourself.

The most important commands are those that allow you to page through the article. To display the next page, press PAGE DOWN, DOWN, or SPACE. To move back one page, press PAGE UP, UP, or **b**. To go to the first page, press **g**. To go to the last page, press **G**.

```
Sun, 23 May 1993 08:22:42       soc.culture.british       Thread 12 of 69
Article 342          Does the Ickenham System work?         63 Responses
gally@pelican.org                  Galahad Threepwood at the Pelican Club

Fred Ickenham tells me that his system is 100% successful -
in certain cases.

I believe that he advises that "The preliminary waggle is of
the essence."

Maybe so, but in my many years as a member of the Pelican Club,
I developed my own methods which, as Sue Brown and Ronnie Fish will
tell you, have been known to meet with some measure of success.

Does anyone else have any thoughts on this matter?

-
Hon. Galahad Threepwood                                Pelican Club, London
          The world will have to wait a hundred years before it
         hears the story of young Gregory Parsloe and the prawns.

<n>=set current to n, TAB=next unread, /=search pattern, ^K)ill/select,
    a)uthor search, B)ody search, c)atchup, f)ollowup, K=mark read,
    |=pipe, m)ail, o=print, q)uit, r)eply mail, s)ave, t)ag, w=post
```

Figure 15-3. *A typical article as displayed by* **tin**

To mark this article as read and go to the next article, press **k** (kill). To mark the entire thread as being read and go to the next thread, press **K**.

If you would like to re-display the article showing the full header, press **^H**. To decode an article using rot-13 (see Chapter 11), press **d**.

To jump to the next unread article, press RIGHT or TAB. To jump to the next thread, press RETURN.

If you want to preserve the article, you have three choices. You can press **m** to mail a copy to someone (including yourself), **o** to print it, or **s** to save it to a file.

To respond to an article, you can press **r** or **R** to send a private reply by mail to the author. The difference between these two commands is that **r** will include the text of the article in your reply.

When you press **r** or **R**, **tin** will set up the header of your message and then start your text editor for you. When you are finished, quit the editor in the regular manner and **tin** will regain control. **tin** will now ask you what you want to do with your reply:

```
q)uit, e)dit, s)end: s
```

You can press whichever letter indicates your choice. As you can see, the **s** (send) choice is the default. So, if you want to send the message—which is usually the case—you need only press RETURN.

A second way to respond to an article is to post a followup article of your own. We will discuss this in the next section.

Finally, when you are finished reading, you can return to the thread selection list by pressing **q** or LEFT. To quit **tin** completely, press **Q**.

Posting an Article

There are three ways to use **tin** to post an article. First, as you are reading an article, you can press **f** or **F** to post a followup. The difference between these two commands is that **f** will include the text of the article in your followup.

When you press **f** or **F**, **tin** will set up the header of your followup and then start your text editor for you. When you are finished, quit the editor in the regular manner and **tin** will regain control. **tin** will now ask you what you want to do with your article:

```
q)uit, e)dit, i)spell, p)ost: p
```

You can press whichever letter indicates your choice. As you can see, the **p** (post) choice is the default. So, if you want to post the article—which is usually the case—you need only press RETURN.

The **i** choice will invoke an interactive spelling checker. However, it will only work if this feature is installed in your system.

The second way to post an article is to press **w** (write) at any time. This tells **tin** that you want to post a regular article (not a followup) to the current newsgroup. Thus, if you are reading the newsgroup selection list, you can post an article to any newsgroup you want by moving to it and then pressing **w**.

After you press **w**, **tin** will ask you for the subject of your article. Type the subject and press RETURN. Everything will now proceed in the regular manner.

From time to time, you may wish to post an article when you are not reading the news (that is, when you are not using **tin**). To do so, you can invoke **tin** with the **-w** (write) option. **tin** will help you post an article and then quit automatically. You will probably also want to use the **-q** (quick start) option to bypass the automatic check for new newsgroups:

```
tin -q -w
```

In general, when you enter a Unix command, you can combine options using a single **-** character, so in this case you can also use:

```
tin -qw
```

When you start **tin** in this manner, it will ask you to enter the name of the newsgroup to which you want to post. After you type the name and press RETURN, everything will proceed in the regular manner. When you are finished posting your article, **tin** will quit automatically.

Commands That Are Always Available

Controlling the Program

q	return to previous level
Q	quit `tin`

Getting Help

h	display summary of commands [help]
H	off/on: show help menu at the bottom of the screen
v	show what version of `tin` you are using
M	display a menu of configurable options

Displaying Information

PAGE DOWN	display the next page
SPACE	same as PAGE DOWN
^D	same as PAGE DOWN [down]
^F	same as PAGE DOWN [forward]
PAGE UP	display the previous page
b	same as PAGE UP [back]
^U	same as PAGE UP [up]
^B	same as PAGE UP [back]

Posting an Article

w	post an article to the current newsgroup [write]
W	display a list of all the articles you have posted

Entering Unix Commands

!*command*	execute the specified Unix command
!	pause `tin` and start a new shell

Figure 15-4. *Commands that are always available*

Commands to Use While Selecting a Newsgroup

Stopping	
q	quit `tin`
LEFT	same as q
Displaying the Selection List	
DOWN	move down one line
j	same as DOWN
UP	move up one line
k	same as UP
num RETURN	go to newsgroup number *num* (for example: 4 RETURN)
1 RETURN	go to the first newsgroup in the list
$	go to the last newsgroup in the list
N	go to the next group with unread news
g	go to specified newsgroup
/	search forward for specified newsgroup
?	search backward for specified newsgroup
Start Reading a Newsgroup	
RIGHT	start reading the current group
RETURN	same as RIGHT
TAB	go to next group with unread news and start reading
n	same as TAB [next]
Controlling the Screen Display	
d	toggle: show newsgroup names/names+descriptions
r	toggle: show all newsgroups/with unread articles only
Controlling Newsgroups	
m	move newsgroup position within the list
s	subscribe to the current newsgroup
u	unsubscribe to the current newsgroup
S	subscribe to all groups that match specified pattern
U	unsubscribe to all groups that match specified pattern

Figure 15-5. *Commands to use while selecting a newsgroup*

Commands to Use While Selecting a Thread to Read

Stopping	
q	return to newsgroup selection list
LEFT	same as q
Moving From One Article to Another	
DOWN	move down one line
j	same as DOWN
UP	move up one line
k	same as UP
num RETURN	go to thread number *num* (for example: 4 RETURN)
1 RETURN	go to the first thread in the list
$	go to the last thread in the list
/	search forward for subject containing a pattern
?	search backward for subject containing a pattern
N	go to next unread thread
P	go to previous unread thread
K	mark thread as read, then go to next unread thread
Start Reading an Article	
RIGHT	start reading the current thread
RETURN	same as RIGHT
TAB	start reading next unread thread
-	return to the last thread that you read
Controlling the Screen Display	
d	toggle: show subject/subject+author
r	toggle: show all threads/unread threads only
Changing Newsgroups	
n	go to next newsgroup
p	go to previous newsgroup
g	go to specified newsgroup
Preserving a Thread	
m	mail the thread to someone
o	print the thread
s	save the thread to a file

Figure 15-6. *Commands to use while selecting a thread to read*

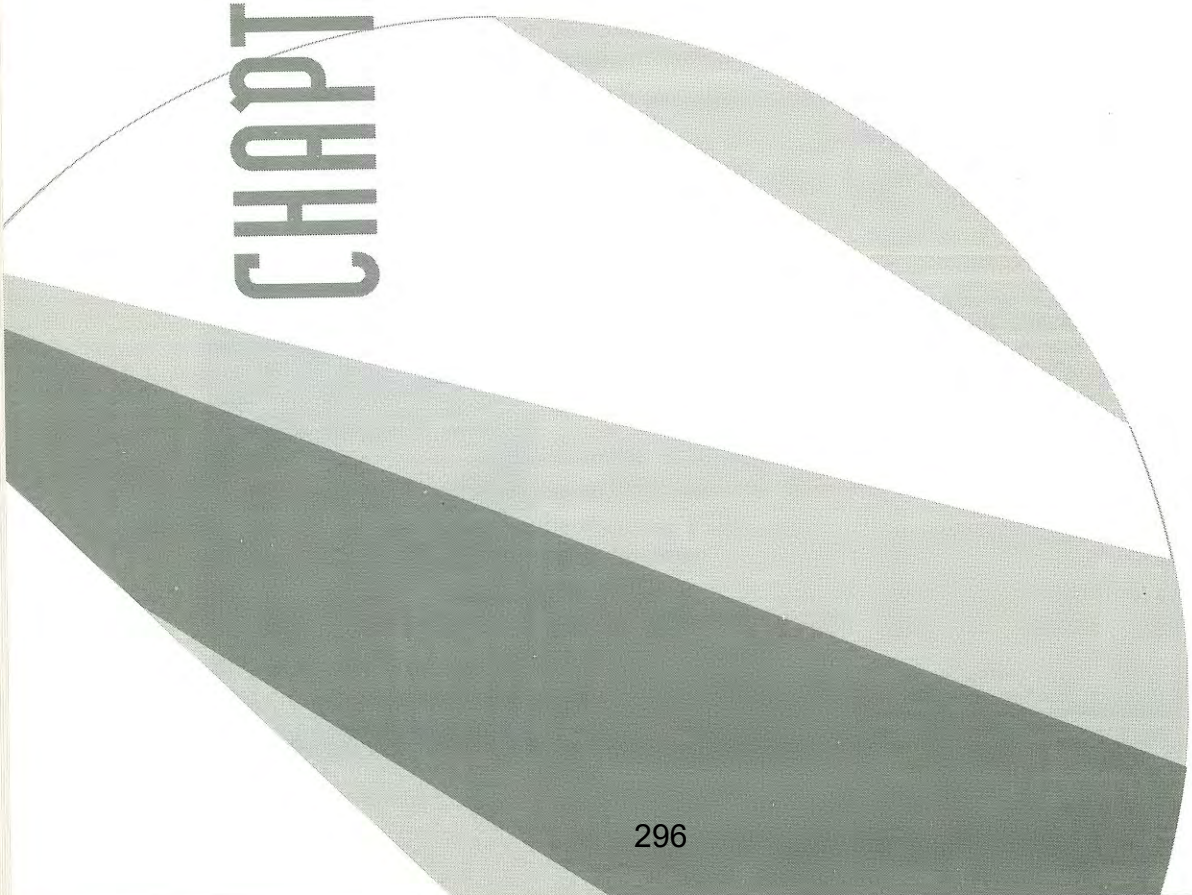
Commands to Use While Reading an Article

Stopping	
q	return to thread selection list
LEFT	same as q
Displaying the Article	
DOWN	display the next page
UP	display the previous page
g	go to the first page of the article
^R	same as g [re-display]
G	go to the last page of the article
$	same as G
^H	re-display article showing header
d	decode the current article using rot-13
/	search forward for specified pattern
?	search backward for specified pattern
Moving to a Different Article	
RIGHT	go to the next unread article
TAB	same as RIGHT
N	same as RIGHT
RETURN	go to the next thread
n	go to the next article
p	go to the previous article
P	go to the previous unread article
K	mark entire thread as read, go to next unread thread
k	mark article as read, go to next unread article
-	return to the last article that you read
Preserving the Article	
m	mail the article to someone
o	print the article
s	save the article to a file
Responding to the Current Article	
F	post a followup
f	same as F, include text of article
R	reply by mail to author
r	same as R, include text of article

Figure 15-7. *Commands to use while reading an article*



CHAPTER 16



Anonymous FTP

One of the most important and most widely used Internet services is Anonymous FTP. This remarkable service lets you copy files from thousands of different computers in all parts of the Internet. These files contain virtually every kind of information that can be stored on a computer.

Would you like a program for your PC or Macintosh? Would you like an issue of an electronic magazine? How about a frequently asked question list from a Usenet discussion group?

Just about any type of information or computer software is waiting for you on the Internet. To access it all—for free—all you have to do is read

this chapter and learn how to use Anonymous FTP. And, if for some reason, you do not have access to FTP, we will show you how to request files by mail.

Once you finish this chapter, be sure to read the related material in Chapters 17 and 18. In Chapter 17, we show you how to use an Archie server to search the vast collection of Anonymous FTP hosts to find specific files. In Chapter 18, we discuss the various types of files that are used to store data for Anonymous FTP retrieval.

Understanding FTP

The term "FTP" comes from File Transfer Protocol, the underlying set of specifications that support Internet file transfer. But what we mean when we refer to FTP is more than a dry set of specifications. FTP is a service that allows us to copy a file from any Internet host to any other Internet host. As such, FTP provides a large part of the spiritual glue that holds the Internet together.

Like most Internet services, FTP uses a client/server system (see Chapter 2). You use a client program, named **ftp**, to connect to a server program on the remote computer. Conceptually, the idea is simple. Using your client program, you issue commands that are sent to the server. The server responds by carrying out whatever commands you sent. For example, you might send a command that requests the server to send you a copy of a particular file. The server responds by sending the file. Your client program receives the file on your behalf and stores it in your directory. (We discuss files and directories later in the chapter.)

As you may remember, when you copy files from a remote computer to your own, we say that you are downloading files. When you copy files from your computer to the remote one, you are uploading files. If you have trouble remembering which is which (and we always do), just imagine the remote computer floating above you in the sky. Up is away from you, down is towards you.

When we talk about FTP, we use the same terminology as for Telnet (see Chapter 7). Your computer is called the local host, and the other computer is called a remote host. Thus, in the language of the Internet, we can say that the **ftp** client program allows you to upload and download files to and from a remote host.

To use the **ftp** program, you enter the **ftp** command and specify the address of the remote host to which you want to connect. Once the program starts, you enter one command after another, to copy files back and forth as you wish. As you use **ftp**, there are a number of commands that you can enter. For example, you can explore the directories on the remote computer and change from one directory to another.

It is common to use the word "FTP" as a verb. For example, you might hear someone say, "You can get a copy of the frequently asked question lists by FTP-ing to **rtfm.mit.edu**. If that computer is too busy, there are a number of other hosts from which you can FTP the same files."

Understanding Anonymous FTP

We have already explained that FTP allows you to transfer files from one Internet computer to another. However, there is one basic restriction: you cannot access a computer unless you have proper authorization. In practice, this means that you must be able to log in to the computer. In other words, you cannot copy files to or from a computer unless you have a userid (account name) and a password.

Anonymous FTP is a facility that lets you connect to a remote host and download files without having to be registered as a user. The system manager sets up a special userid named **anonymous** that anyone, anywhere on the Internet, is allowed to use.

Hint

You cannot use Anonymous FTP with every Internet host, only with those that have been set up to offer this service.

When you use the **ftp** program to connect to an Anonymous FTP host, it works the same way as regular FTP, except that when you are asked for a userid you enter **anonymous**. When you are asked for a password, you enter your mail address or name. This is so the keepers of the system can keep track of who is accessing their files. (For a discussion of Internet addresses, see Chapter 4.)

Hint

The **anonymous** userid can only be used with FTP. You cannot connect to an Anonymous FTP host using Telnet (see Chapter 7) and use the **anonymous** userid to log in as a regular user.

When a system manager sets up a computer to be an Anonymous FTP host, he or she designates certain directories as being open to public access. The rest of the directories are off limits. Thus, it is perfectly safe for an organization to offer public access to outside users. As an extra security measure, most Anonymous FTP hosts allow you to download files, but not upload. That is, you can copy all the files you want from the remote host, but you cannot copy files to the host.

If an Anonymous FTP host does allow uploads, you may be required to copy all new files to a single designated upload directory. Later, after the system

manager has had a chance to check out the new files, he or she will move them to one of the public download directories. In this way, remote users are protected against people who might upload troublesome files, such as programs with viruses.

Hint

Most system managers do not have the time to be eternally vigilant. When you download programs, please remember to take normal precautions. For example, before you run a program on your PC, check it with a virus scanning program.

As an Internet user, you can use FTP to copy files between any two Internet hosts. Realistically, though, most people only have a single Internet account and FTP is mostly used to download public files. Thus, in this chapter, we will concentrate on what you need to know to download using Anonymous FTP.

As you will see, the **ftp** client program has a large number of commands that you can use, and some of those commands have variations. We will include a list summarizing all the important commands and what they do, and we will show you how to display help information as you are working. But we will not discuss every command in detail. Rather, we will spend our time on what you need to know to use Anonymous FTP.

If you are using a Unix system and you would like more detailed information about the **ftp** program and its commands, you can display the **ftp** entry in the online manual. Use the following command:

```
man ftp
```

We discuss the **man** command and how to use it in Chapter 11.

The Importance of Anonymous FTP

At first, Anonymous FTP sounds useful, but not really all that special. Okay, so anyone on the Internet can access files on any Anonymous FTP host. What's the big deal?

The big deal is that the world of Anonymous FTP is huge. There are thousands of Anonymous FTP hosts and countless files that you can download for free. Virtually every type of information and every type of computer program

is available somewhere on the Internet. Many people and organizations have generously donated disk space and computing facilities as well as their own time to make these files available. Why? Because, as we explained in Chapter 1, the tradition on the Internet is one of sharing.

Until you start using Anonymous FTP, this may not sound like much, but it is. Indeed, we can solemnly assure you that Anonymous FTP is one of the most significant inventions in the history of mankind.

If you are new to the Internet, we don't blame you one bit for thinking that such a statement is the rankest hyperbole. Still, what we say is absolutely true, for three important reasons.

First, we have already said that Anonymous FTP allows you to download virtually any type of information. Until you are an experienced Internet user, it is difficult to appreciate how important this is. Anonymous FTP provides access to the largest library of information ever accumulated. Moreover, it is a library that is always growing, never closes, covers every conceivable topic and, best of all, is free. To get a feeling for the variety of information, take a moment now to skim through the catalog in the last part of this book. Notice how many of the items are available via Anonymous FTP.

Hint

With so many different Anonymous FTP hosts and so many files, how do you know where to find a particular file? You use a service called Archie. Archie will search a special FTP database and find all the hosts that contain the file you need. For an explanation of how to use Archie, see Chapter 17.

Second, Anonymous FTP is the principal way in which software is distributed on the Internet. The reason that the Internet can exist at all is that people use programs that provide standardized services using standardized protocols. Many of these programs are distributed via Anonymous FTP and, hence, are available to anyone who wants to set up an Internet host.

For example, in order for you to access the Usenet discussion groups, your system manager must have installed the Usenet software on your computer. Where does your system manager acquire this software? Anonymous FTP.

The final reason why Anonymous FTP is so important is that it is used to archive and disseminate the technical information that defines the Internet itself. As we explained in Chapter 2, the Internet is based on a large number of protocols and conventions. Each such protocol is explained in a technical publication called a *request for comment* or *RFC*. Don't take the name too literally. Think of an RFC as a technical explanation of how something is supposed to work. RFCs are also used to offer generally useful information on a specific topic.

Each RFC is given a number and is made freely available to anyone who wants to read it. For example, RFC #1325 is a long list of answers to questions that are commonly asked by new Internet users. (This RFC is similar to the frequently asked question lists that we discussed in Chapter 9.)

Hint

If you would like to read RFC #1325, use Archie (see Chapter 17) to search for the file named **rfc1325.txt**. Once you have found an Anonymous FTP host that has this file, use the techniques described in this chapter to FTP to the host and download the file.

If you would like to download an index of all the RFCs, use Archie to search for a file named **rfc-index.txt**.

Starting the ftp Program

To download or upload files from a remote host, you use the **ftp** program. This program acts as a client and connects to the FTP server on a remote host. Once the connection is made, you will be asked to specify a userid and password. You can then enter whatever **ftp** commands you want.

There are two ways to start the **ftp** program. In this section, we will show you how it is done most of the time. In the next section, we will show you an alternate method.

To start **ftp**, enter the name of the command followed by the address of the remote host to which you want to connect. For example, say that you want to download files from the computer named **rtfm.mit.edu**. Enter the command:

```
ftp rtfm.mit.edu
```

Hint

As we explained in Chapter 4, all Internet hosts have an official address known as an IP address. This address consists of several numbers separated by periods. For example, the official IP address of **rtfm.mit.edu** is **18.70.0.224**.

Some systems have trouble dealing with certain standard addresses. If you encounter such a problem with **ftp**, try using the IP address. For example, either of the following commands will connect to the same host:

```
ftp rtfm.mit.edu
ftp 18.70.0.224
```

For more information about IP addresses and Internet addresses in general, see Chapter 4.

When the **ftp** program starts, it will initiate a connection to the remote host that you specified. Once the connection is made—which might take a few moments if the host is far away—you will see a message like the following:

```
Connected to CHARON.MIT.EDU.
220 charon FTP server (Version 6.6 Wed Apr 14 21:00:27 EDT 1993) ready.
Name (rtfm.mit.edu:harley):
```

The first line of this message shows us that we have made the connection. Notice that the name of the computer we have connected to is really **charon.mit.edu**. This is because the name **rtfm.mit.edu** is actually an alias for **charon.mit.edu**.

Such aliasing is fairly common in the world of Anonymous FTP because it allows us to use easily remembered names. It also gives system managers the flexibility to change computers whenever they want without confusing people. For example, if it should become necessary for the **rtfm** system manager to use a different computer for Anonymous FTP, he does not have to inform everybody of the change. All he has to do is ensure that the address **rtfm.mit.edu** is aliased to the new computer.

It is especially common to see names that begin with **ftp**. For example, the Electronic Frontier Foundation has an Anonymous FTP host named **ftp.eff.org**. (The EFF is a public-service organization dedicated to "the pursuit of policies and activities that will advance freedom and openness in computer-based communications". As part of its work, the EFF maintains an Anonymous FTP host that contains a wealth of interesting information, including electronic magazines.)

The public name of the EFF's Anonymous FTP host is **ftp.eff.org**. However, when you FTP to this address, you will see that it is actually aliased to another computer. At the time we wrote this chapter, the real computer happened to be **krager.eff.org**.

Now, to return to our example, the second line we saw was:

```
220 charon FTP server (Version 6.6 Wed Apr 14 21:00:27 EDT 1993) ready.
```

This tells us the name of the FTP server and the version of the FTP software that this server is using.

Hint

Notice that the message starts with the number **220**. All messages from the FTP server start with such numbers, and there is no way to get rid of them. However, the numbers are not important, and you can ignore them.

One nice thing about the numbers is that they show you exactly which messages are coming from the remote server. Lines that do not begin with a number are from the **ftp** client program.

Finally, let us look at the last line of the message:

```
Name (rtfm.mit.edu:harley):
```

This is a request from our **ftp** client program, asking us what userid we want to use to log in to the computer named **rtfm.mit.edu**. As it happens, we are currently logged in to the local host as **harley**. The **ftp** program knows this and suggests that we may want to use the same userid on the remote system. That is why you see the name **harley** in parentheses.

If you press RETURN, the **ftp** program will use this userid as a default and send it to the remote host. However, in this case (and most of the time), you will want to log in as **anonymous**. Simply type this name and press RETURN:

```
anonymous
```

You will now see:

```
331 Guest login ok, send e-mail address as password.
Password:
```

The FTP server has approved the userid **anonymous**. You are now being asked to enter your mail address as a password. Out of courtesy, it is a good idea to honor this request. Indeed, some FTP servers will not let you log in if your password does not look like a valid address.

Hint

Some FTP servers will examine your password and decide if it looks like your real mail address. If not, you may be denied access to certain public directories (and you may not even know that this is happening).

Once you have entered an approved userid and password, you will see a message like the following:

```
230 Guest login ok, access restrictions apply.
ftp>
```

This means that you are officially logged in and can use the Anonymous FTP facility.

The second line, **ftp>**, is a prompt from your **ftp** client program. Whenever you see this prompt, you can enter one of the **ftp** commands. We will discuss the various commands later in the chapter, and at that time we will show you a full Anonymous FTP session in which we download a file.

For now, we just want to mention two commands. To display a list of all the **ftp** commands, enter the **?** command. To end the FTP session, use the **quit** command.

The last point we want to cover is what to expect if your FTP client is unable to connect to the remote host. There are three ways in which this might happen. First, the FTP service may be temporarily unavailable. In such cases, you will see:

```
ftp: connect: Connection refused
```

Second, the network connection to the remote host may be inoperative. This might be a problem with the network to which the host is connected, or it might be that the host computer itself is not working. In such cases, you will see a message like:

```
ftp: connect: Host is unreachable
```

The best thing to do is try again later.

Finally, there may be a problem with the address that you specified. For example, say that you want to download files from **rtfm.mit.edu**, but you accidentally enter the wrong address:

```
ftp rtff.mit.edu
```

You will see a message like the following:

```
rtff.mit.edu: unknown host
ftp>
```

At this point, you can enter the name of another host. (We will explain how to do this in the next section.) Otherwise, you can use the quit command to stop the program.

Hint

The **ftp** message "unknown host" can be misleading. There are many reasons why your FTP client might not be able to make a remote connection. The two most common are:

- You spelled the address of the computer wrong.
- You specified the name of a computer that is not on the Internet.

A Second Way to Start the ftp Program

In the previous section, we mentioned that there are two ways to start the FTP client program. The first way is to enter the **ftp** command along with the address of the remote host. For example:

```
ftp rtfm.mit.edu
```

The second way is to start **ftp** without specifying a host. Simply enter:

```
ftp
```

The program will start, but will not make a connection. Instead, you will see:

```
ftp>
```

This is the **ftp** prompt. It means that the program is waiting for you to enter a command. To connect to a remote host, type **open**, followed by the address of the host. For example:

```
open rtfm.mit.edu
```

The connection will be made just as if you had specified the address when you entered the **ftp** command.

At the end of the previous section, we gave an example in which an **ftp** command had a bad address. In the example, the remote host was named **rtfm.mit.edu**, but we mistakenly entered:

```
ftp rtff.mit.edu
```

What happens in such a case is that the **ftp** program tries to make the connection. When it can't, it gives up and displays its prompt, waiting for you to enter a command. In this case, you would see:

```
rtff.mit.edu: unknown host
ftp>
```

You can now enter:

```
open rtfm.mit.edu
```

If this address doesn't work, you can try another one. If you decide to give up, enter:

```
quit
```

This will stop the **ftp** program.

Summary of Starting and Stopping ftp

There are two ways to start the **ftp** program. Either enter the command with the address of a remote host:

```
ftp rtfm.mit.edu
```

or enter the command by itself:

```
ftp
```

and then, at the **ftp>** prompt, enter an **open** command:

```
open rtfm.mit.edu
```

To stop **ftp**, wait for the **ftp>** prompt and enter the **quit** command:

```
quit
```

Directories and Files

Once you start messing around with Anonymous FTP, you will find that you need to know something about files and directories. Since most of the Anonymous FTP hosts are Unix computers, you will need to know about the Unix file system.

You may remember that, in Chapter 1, we told you it is a good idea to learn some Unix. We pointed out that even if you do not use Unix yourself, many of the computers that you will access are Unix computers. This is especially true when you are using Anonymous FTP.

In this section, we will give you the short, 75-cent tour of the Unix file system. If you want more information, the best idea is to consult a good Unix book. The one we recommend is *A Student's Guide to Unix*, by Harley Hahn (McGraw-Hill , 1993).

Within Unix, the definition of a *file* is a highly generalized one. A file is considered to be any source of input or any target of output. For our purposes,

this definition is too technical, so let us be a bit more informal: a file is a collection of data.

Files can hold any type of information that can be stored on a computer: programs, documents, images, and so on. Most files are stored on disks (hard drives), but you will also hear of files stored on CDs, tapes, and floppy diskettes.

At the time a file is created, it is given a name. Whenever you want to do something with a file, you must refer to it by its name. For example, when you want to download a file using Anonymous FTP, you must be able to tell your FTP client the name of the file you want.

Unix file names can consist of letters, numbers, and certain punctuation characters. For example, earlier in the chapter we referred to two files named **rfc1325.txt** and **rfc-index.txt**.

Unix distinguishes between uppercase (capital) letters and lowercase (small) letters. This means that when you use a file name, you must use the exact letters. For example, it would be wrong to refer to the file above as **Rfc1325.txt**. To Unix, the letters **R** and **r** are completely different, so **rfc1325.txt2** and **Rfc1325.txt** are two completely different names.

For the most part, Unix file names consist of all lowercase letters. However, there will be times when you encounter a file name that has uppercase letters. When you do, be sure to type the name exactly.

Since most computers can store vast numbers of files, Unix allows us to organize them into collections called *directories*. A directory is itself a special kind of file whose only purpose is to act as a repository for other files.

When you connect to a remote Anonymous FTP host, you will usually find a large number of directories. In order to download a file, you must be able to specify the name of the directory in which the file lies, as well as the name of the file itself. Here is an example.

In Chapter 9, we explained that many Usenet newsgroups have lists of questions and answers that cover the basic topics discussed in that newsgroup. These lists are called FAQ (frequently asked question) lists. The computer **rtfm.mit.edu** acts as a Usenet archive by storing copies of the FAQ lists for people to access via Anonymous FTP.

One of the more interesting lists is from the newsgroup that discusses urban legends (**alt.folklore.urban**). The name of this file is **folklore-faq**, and it is found in the directory named **/pub/usenet/news.answers**. To make sense of this directory name, all you need to understand is how Unix organizes directories.

Being able to collect files into directories is important, as it allows us to organize large numbers of files. However, many Unix systems have thousands of files and merely collecting them into directories is not enough: we need more levels of organization.

The solution is to collect the directories themselves into other directories. This allows us to form a hierarchy in which some directories contain other directories.

For example, on the **rtfm.mit.edu** host, all the public directories are kept in a directory named **pub**. Within **pub**, all the directories that contain Usenet information are kept in a directory named **usenet**. Within the **usenet** directory, there are files and directories that contain information relating to the various newsgroups.

Now, it happens that there is a special newsgroup named **news.answers** whose purpose is to act as a repository for all the FAQ lists that are posted to the various other groups (see Chapter 9). Thus, on the **rtfm.mit.edu** host, the **usenet** directory contains a directory named **news.answers** that holds all the FAQ lists. That is why the file named **folklore-faq** is found within the directory named **/pub/usenet/news.answers**.

When a directory contains another directory, we call the second one a *sub-directory*. Sometimes, the first directory is called the *parent directory*, while the sub-directory is called the *child directory*.

Each Unix computer has one main directory that contains all the sub-directories and files on the system (at least, indirectly). This main directory is called the *root directory*. Within the root directory are a number of sub-directories; within these sub-directories are other sub-directories; and so on.

In our example, **pub** is a sub-directory of the root directory; **usenet** is a sub-directory of **pub**; **news.answers** is a sub-directory of **usenet**; and within **news.answers** lies the file **folklore-faq**.

If you were to draw a diagram of such a system, it would look like a tree with branches and sub-branches growing from one main trunk, hence the name "root directory" for the main trunk.

When we write the name of a sub-directory, we start at the root directory and show each directory that we must pass through in order to reach the ultimate branch. To understand such names, you only need to remember two things.

First, the root directory is indicated by a single / (slash) character. We do not actually spell the name "root". Second, we use the same / character to divide one directory name from another. Hence, the name **/pub/usenet/news.answers**. Literally, it means "Start from the root directory, then go to the **pub** sub-directory, then go to the **usenet** sub-directory, and finally, go to the **news.answers** sub-directory."

As you use a Unix system, you can think of yourself as sitting somewhere in the directory tree. Whichever directory you are in is called your *working directory* or *current directory*. By using the appropriate commands, you can move from one directory to another. (This is like moving from one branch of the tree to another.) As you will see, it is often convenient to move to the directory that contains the file you want before you download it.

For convenience, we often describe the location of a file by tacking its name onto the end of a sequence of directories. For example, say that you wanted to tell a friend where to find the FAQ list for the Usenet newsgroup that discusses

urban folklore. All you have to do is tell him to FTP to **rtfm.mit.edu** and copy the file **/pub/usenet/news.answers/folklore-faq**.

A description like this—in which names are separated by / characters—is called a *path* or a *pathname*. Thus, in order to FTP a file anywhere on the Internet, all you need to know is (1) the name of an Anonymous FTP host that has the file, and (2) the pathname of that file. If you take a look at the catalog in the last part of this book, you will see we describe many Internet resources in just this way.

An Overview of the ftp Commands

Once you have entered the **ftp** command and established a connection with a remote host, you will see the prompt:

```
ftp>
```

At this point, you can enter an **ftp** command (of which there are many). The FTP client program will send whatever command you enter to the FTP server, which will carry out your request. The idea is to enter one command after another until you have achieved your goal (say, to download a file). Then enter the **quit** command to terminate the FTP session.

At any time, you can display a list of all the **ftp** commands by entering **?** or **help** (either will do). Figure 16-1 contains a typical response from such a command. (Do not worry if your **ftp** client does not have all these commands. It will have the most important ones.)

If you want to see a one-line summary of a particular command, enter **?** (or **help**) followed by the name of the command. For example, if you enter:

```
? quit
```

you will see:

```
quit    terminate ftp session and exit
```

Notice how many commands there are in Figure 16-1. The number of commands that you see will vary, depending on what version of the FTP software you are using.

You might wonder, do I really have to learn all these commands? The answer is no. For normal Anonymous FTP sessions, all you need to know are the

At the **ftp>** prompt, you can enter **?** or **help** to display a summary of all the possible commands that are recognized by the FTP server. Here is such a summary:

```
Commands may be abbreviated.  Commands are:

!               cr              macdef          proxy           send
$               delete          mdelete         sendport        status
account         debug           mdir            put             struct
append          dir             mget            pwd             sunique
ascii           disconnect      mkdir           quit            tenex
bell 8          form            mls             quote           trace
binary          get             mode            recv            type
bye             glob            mput            remotehelp      user
case            hash            nmap            rename          verbose
cd              help            ntrans          reset           ?
cdup            lcd             open            rmdir
close           ls              prompt          runique
```

Figure 16-1. *A list of all the* **ftp** *commands*

commands summarized in Figure 16-2. We will discuss these commands in the following section.

Hint

To stop an **ftp** command as it is executing, press CTRL-C.

The Basic ftp Commands

We can divide the **ftp** commands into several groups. First, there are the basic commands. We have already discussed **quit**, **?**, and **help** (which is the same as **?**).

The other basic command is **!** (exclamation mark). This command is used on Unix systems to enter a regular command to your local computer. Simply type the command after the **!** character and then press RETURN. The **ftp** program will put itself on hold and send the command to your Unix shell to be executed. Once the command is finished, the **ftp** program will regain control and redisplay its prompt.

Here is an example. We want to use the Unix **date** command to display the time and date on your local system. At the **ftp>** prompt, we enter **!date**. After the **date** command displays its output, we are returned to a **ftp>** prompt:

```
ftp> !date
Thu Jun 17 23:11:19 PDT 1993
ftp>
```

Basic Commands

`quit`	close connection to remote host, stop `ftp` program
`?`	display a list of all the `ftp` commands
`? command`	display one-line summary of the specified command
`help`	display a list of all the `ftp` commands
`help command`	display one-line summary of the specified command
`!`	local: pause `ftp` and start a shell
`! command`	local: execute specified shell command

Connecting

`open [host]`	establish connection to specified computer
`close`	close the connection to remote host, stay in `ftp`
`user [name [password]]`	set user name

Directories

`cd [directory]`	remote: change to specified directory
`cdup`	remote: change to parent directory
`dir [directory [local-file]]`	remote: display a long directory listing
`lcd [directory]`	local: change directory
`ls [directory [local-file]]`	remote: display a short directory listing
`pwd`	remote: display name of current directory

Transferring Files

`get [remote-file[local-file]]`	download one file
`mget [remote-file...]`	download multiple files

Setting Options

`ascii`	(default) set file type to ASCII text file
`binary`	set file type to binary file
`hash`	yes/no: show # for each data block transferred
`prompt`	yes/no: prompt for multiple file transfers
`status`	display current status of options

Figure 16-2. *Summary of the most useful* **ftp** *commands*

If you would like to enter more than one Unix command, you can use the ! character by itself:

```
ftp> !
```

The **ftp** program will put itself on hold and start a new shell. You can now enter as many Unix commands as you want. When you are finished, terminate the shell and the **ftp** program will regain control. With most shells, you would press CTRL-D to terminate the shell. If this does not work, try the **exit** command.

The Connection ftp Commands

The next category of **ftp** commands are those that control the connection to the remote host: **open, close,** and **user**. We have already discussed the **open** command which you can use to establish an FTP connection.

The **close** command will terminate an FTP connection without quitting the **ftp** program. You can use **close** when you want to close one connection and then open another one.

If you connect to a remote host successfully, but there is something wrong with your userid or password, you may or may not lose your connection. (This depends on the FTP server.) However, if you do not lose the connection, you will not be able to do anything until you specify a valid userid and password. To do so, you can use the **user** command. Simply enter:

```
user
```

The **ftp** program will ask you to enter a userid and then a password. If you want, you can specify the information directly:

```
user anonymous harley@fuzzball.ucsb.edu
```

In this case, we are specifying a userid of **anonymous** and a password of **fuzzball.ucsb.edu**.

The Directory ftp Commands

The third group of **ftp** commands are those that you can use to move from one directory to another and to display the contents of a directory. As we explained, whatever directory you are in is called your working directory (or current directory). Once you establish an FTP connection, you have two working directories to keep track of: one on the remote host and one on your local computer.

With most Anonymous FTP hosts, you are automatically placed in the root (top-level) directory to start. To move to another directory, use the **cd** (change directory) command. Type the command name, followed by the name of the directory you want to move to, and then press RETURN. For example, if you want to move to the directory named **/pub/usenet/news.answers**, enter:

```
cd /pub/usenet/news.answers
```

If you understand the Unix file system, you can move one directory at a time by using separate **cd** commands. For example:

```
cd pub
cd usenet
cd news.answers
```

Unfortunately, the techniques that you need to use the **cd** command in this way are beyond the scope of the book. Please refer to a Unix book for more details.

At any time, you can display the name of your remote working directory by using the **pwd** (print working directory) command. For example, if you enter:

```
pwd
```

you will see a message similar to:

```
257 "/pub/usenet/news.answers" is current directory.
```

(Remember, messages from the FTP server always start with a number.)

At times, you may see a directory name that is different from what you expect. For example, on the **rtfm.mit.edu** system, if you move to the **/pub/usenet/news.answers** directory and then enter a **pwd** command, you will actually see:

```
257 "/pub/usenet-by-group/news.answers" is current directory.
```

Do not be confused. All this means is that the system manager has given the second level directory two different names: **usenet** and **usenet-by-group**. In such cases, you can use whichever name you want. The name **usenet-by-group** is more informative, but the name **usenet** is a lot easier to type.

Hint

In Unix, the word "print" is often used to mean "display". Thus, the job of the **pwd** (print working directory) command is to display the name of your working directory. This tradition dates back to the earliest days of Unix when terminals actually printed their output on paper.

The working directory on your local computer will be whatever directory you happened to be in when you entered the **ftp** command. When you download files, this is the directory into which they will be placed. If you want your files to be placed into a different directory, you can use the **lcd** (local change directory) command to move to a different directory before you start the downloading. For example:

```
lcd faq-files
```

This command changes your local working directory to **faq-files**. Again, you will find it helpful to read a Unix book for more information. In particular, you should learn how to make your own directories and how to move from one directory to another.

There is no **ftp** command to display the name of your local working directory. However, it is possible to do this by using the **!** command to send a **pwd** command to your local computer:

```
!pwd
```

Hint

Before you start an Anonymous FTP session, decide which directory on your local computer you want to use to store the downloaded files. Use the Unix **cd** command to move to that directory *before* you enter the **ftp** command. That way, you will not have to use a **lcd** command to ensure that your downloaded files are placed in the right directory.

To display the contents of a directory on the remote host, you can use two different commands. The **ls** (list) command will display the names of all the files in the directory. Here is some typical output. It shows the names of some of the files in the **news.answers** directory that we discussed above:

```
esperanto-faq
feminism
finding-addresses
finding-sources
fleas-ticks
folklore-faq
fonts-faq
```

The **dir** command will display a longer listing. Along with the file name, you will also see extra information. Here is some typical output:

```
-rw-rw-r--  8 root   3   27120 Jun  2 01:24 esperanto-faq
drwxrwxr-x  2 root   3     512 Jun 12 00:07 feminism
-rw-rw-r-- 14 root   3   28880 Jun 12 03:37 finding-addresses
-rw-rw-r-- 12 root   3   41939 Jun 16 04:04 finding-sources
-rw-rw-r-- 10 root   3   41533 Jun 16 03:30 fleas-ticks
-rw-rw-r--  8 root   3   84701 Jun 15 03:33 folklore-faq
drwxrwxr-x  2 root   3     512 Jun 18 01:46 fonts-faq
```

If you have some experience with the Unix file commands, you will recognize this output as being from the **ls -l** command. Otherwise, don't worry about it. You only need to understand four things.

First, the character at the far left tells whether the line describes a directory or a file. A **d** character indicates a directory while a **-** (hyphen) character indicates a file. In this example, we have two directories and five files.

At the far right of each line, we see the name of the file or directory. To the left of the name, the time and date show you when the file or directory was updated.

Finally, the number to the left of the time and date shows the size of the file in bytes (characters). This number is not meaningful for directories.

Thus, we can see that **folklore-faq** is a file, it was last updated on June 15 at 3:33 AM, and is 84,701 characters long. You can ignore the rest of the information. You will find that it is often convenient to download a copy of a long directory listing to a file on your local computer. To do so, simply specify the name of the directory followed by the name of a file. For example:

```
ls /pub/usenet/news.answers ls.list
dir /pub/usenet/news.answers dir.list
```

Each of these commands generates a directory listing of the specified directory. The first command downloads its output to a file on your local computer named **ls.list**. (Of course, you can choose whatever names you want.)

When you use this form of the **ls** and **dir** commands, you must always specify a directory name and a local file name. If you want to download a listing of the working directory, you can use a directory name of **.** (a single period character). In Unix, a single **.** character stands for your working directory. For example:

```
ls . ls.list
dir . dir.list
```

The File Transfer ftp Commands

There are two commands that you can use to download files (that is, to copy files from the remote host to your computer). These commands are **get** and **mget**.

The **get** command allows you to download one file at a time. The **mget** (multiple get) command allows you to download more than one file at a time.

To use **get**, simply specify the name of the remote file followed by the name that you want to give the file on your local computer. For example, say that you have established an Anonymous FTP session with the **rtfm.mit.edu** computer that we mentioned earlier. You would like to download the **folklore-faq** file. You want the file to be named **urban-legends** on your computer.

First, move to the directory that contains the file:

```
cd /pub/usenet/news.answers
```

Now, enter the **get** command to download the file:

```
get folklore-faq urban-legends
```

You will see the following messages:

```
200 PORT command successful.
150 Opening ASCII mode data connection for folklore-faq (84701 bytes).
```

At this point, the file is being copied to your computer. Once the copy is complete, you will see:

```
226 Transfer complete.
local: urban-legends remote: folklore-faq
86113 bytes received in 17 seconds (4.9 Kbytes/s)
ftp>
```

You can now enter another command. (By the way, don't worry about the fact that the file is 84,701 bytes long, but we received 86,113 bytes. The extra bytes have to do with how certain characters are encoded when you copy a text file. Rest assured that the actual downloaded file will be exactly the right size.)

If you use the **get** command with only a single file name, the **ftp** program will use this name for the new file on your local computer. For example, if you enter:

```
get folklore-faq
```

the downloaded file will automatically be named **folklore-faq**.

Hint

If you already have a file with the same name on your local computer, the existing file will be replaced. Once a file is replaced, there is no way to get it back, so be careful.

If you are using a Unix system, there is a way to send the remote file as input to a Unix command on your local computer, rather than to a local file. Instead of a second file name, you type a | (vertical bar, called the *pipe* symbol), followed by the name of the command.

For example, suppose that you want to read a remote file named **README**. One way is to download it to a local file, stop or pause the FTP session, read the file, delete the file, and then resume the FTP session. Alternatively, you can read

the remote file by sending it directly to a paging program (such as **more**) on your local computer:

```
get README |more
```

Notice that you cannot put a space after the | character (like you can with a regular Unix command). This is because **get** expects only two words: the remote file name (in this case, **README**) and the name of the local target (in this case, **|more**).

The **mget** (multiple get) command is used when you want to download more than one file at the same time. Type the command, followed by the names of the files you want to download. For example:

```
mget finding-addresses finding-sources folklore-faq
```

When you use **mget**, you cannot specify alternate names, so the files will be given the same names when they are copied to your local computer.

The **mget** command will transfer one file at a time. Before transferring a file, **mget** will display the file name and ask you for confirmation. For example:

```
mget finding-addresses?
```

At this point you can type either **y** (yes) or **n** (no) and press RETURN.

Hint

Remember, you can interrupt any **ftp** command, including **get** and **mget**, by pressing CTRL-C.

The nice thing about **mget** is that you can specify the name of a directory and **mget** will process each file in the directory. Similarly, if you know how to use the Unix wildcard characters, you can download all the files whose names match a particular pattern.

If you do not know about wildcard characters, we will mention briefly that an * (asterisk) character matches zero or more characters, and a ? (question mark) character matches any single character. For example, to download all the files in the remote working directory whose names begin with the letters **fi**, you can use:

```
mget fi*
```

For more information on using such patterns, consult a Unix book. (Or experiment using the **ls** command; for example, try **ls fi*** to list the names of all the files that begin with **fi**.)

Hint

In this chapter, we have concentrated on downloading because that is what you usually do with Anonymous FTP. However, the **ftp** program also has commands to upload. They are **put** (to upload a single file) and **mput** (to upload more than one file at a time). These commands use the same format as **get** and **mget**.

For more information, see the documentation for your local **ftp** program. If you are using a Unix computer, you can display the **ftp** entry in the online manual by entering the command **man ftp**.

Setting ftp Options

Within the **ftp** program, there are several commands that you can set to control the downloading operation. The ones we will talk about are **binary**, **ascii**, **hash**, **prompt**, and **status**.

In Chapter 5, we explained the difference between text files and binary files. Briefly, a text file (also called an ASCII file) holds ordinary characters: letters, numbers, punctuation, and so on. The files that we used as examples in this chapter are all text files.

A binary file contains information that is not textual. For example, if you want to download files that contain pictures, you will be dealing with binary files.

Be default, the **ftp** program assumes that it is working with text files. If you want to download binary files, you should tell the program before you enter the **get** or **mget** command.

The **binary** command tells the **ftp** program that you will be downloading binary files. If you want to switch back, the **ascii** command indicates that you will be downloading text files. When you use one of these commands, we say that you are setting the *representation type*.

For example, say that you are about to download some files that contain pictures. Before you do, you enter the command:

```
binary
```

When you enter this command, you will see the following message:

```
200 Type set to I.
```

The I stands for "image". (Don't worry about it.)

If you use the **binary** command and then download a text file, it will work just fine (although it may be a bit slower). It's only when you are copying binary files that you should be exact. With some remote hosts, a binary file will not be copied properly unless you have set the representation type to **binary**.

Hint

You will often encounter files whose names end in **.Z** or **.tar**. These files use special formats that we will discuss in Chapter 18. (A **.Z** file is compressed; a **.tar** file contains a collection of files called an archive.) Such files are always binary files.

Files whose names end in **.txt** are always text files.

If you are not sure what type of file you are downloading, set **binary** just to be safe.

The next option we want to mention is **hash**. This tells the **ftp** program to display a # character (sometimes called a hash mark) after each data block is transferred. This allows you to watch the progress when you are downloading a large file. The size of a data block depends on the nature of your FTP connection, but you will be told what it is before the downloading starts.

To turn on the **hash** option, just enter:

```
hash
```

To turn it off, enter the same command again.

The last option we want to mention is **prompt**. As we explained in the previous section, the FTP client will query you about each file whenever you use **mget**. If you turn the **prompt** option off, **mget** will automatically transfer each file without asking your permission.

To turn off the **prompt** option, just enter:

```
prompt
```

To turn it back on, enter the same command again.

Finally, if you want to display the current setting of all the options, you can use the **status** command:

```
status
```

When you do, you will see many options, most of which you can safely ignore.

A Typical Anonymous FTP Session

Figure 16-3 contains the full listing of a typical Anonymous FTP session. The commands that we entered are in boldface.

In this example, we connect to **rtfm.mit.edu**. We use a userid of **anonymous** and a password of our mail address. (As with all such systems, the password is not echoed as we type it.)

```
% ftp rtfm.mit.edu
Connected to CHARON.MIT.EDU.220 charon FTP server
 (Version 6.6 Wed Apr 14 21:00:27 EDT 1993) ready.

Name (rtfm.mit.edu:harley): anonymous
331 Guest login ok, send e-mail address as password.
Password:230 Guest login ok, access restrictions apply.

ftp> cd /pub/usenet/news.answers
250 CWD command successful.

ftp> hash
Hash mark printing on (8192 bytes/hash mark).

ftp> get folklore-faq
200 PORT command successful.
150 Opening ASCII mode data connection for folklore-faq (84701 bytes).
############
226 Transfer complete.local: folklore-faq remote: folklore-faq
86113 bytes received in 17 seconds (4.9 Kbytes/s)

ftp> quit
221 Goodbye.
```

Figure 16-3. *A sample Anonymous FTP session*

We then change to the **/pub/usenet/news.answers** directory and turn the **hash** option on. Next, we use **get** to download the file named **folklore-faq**.

Once the downloading is complete, we enter the **quit** command to close the connection to **rtfm.mit.edu** and end the session.

Finding Your Way Around a New Computer

When you FTP to a new computer, it can sometimes take a while for you to find your way around. To orient yourself, start by looking for certain files and directories.

First, look for a directory called **pub**. Most Anonymous FTP hosts contain such a directory to hold all the public files and sub-directories.

Next, in each directory you use, look for one or more of the following files:

```
README
index
ls-lR.Z
ls-ltR.Z
```

If the **README** or **index** files exist, download and read them before you download other files. The **README** file will contain general information, while the **index** file will have a description of what is available.

Hint

The first time you use a new Anonymous FTP host, look for a file named **README** in the root directory.

The **ls-lR.Z** and **ls-ltR.Z** files will contain a detailed listing of the public files and directories. The name comes from the Unix **ls** (list) command that is used to generate such a listing. We won't go into the details except to say that the **ls-ltR.Z** file is sorted by time, showing the most recently modified files first. The **ls-lR.Z** file is sorted by regular, alphabetical order. If you want to look at such a listing, you only need to download one of these files.

The **.Z** characters at the end of these file names indicate that the files are stored in a special compressed format. When you download such files, you must

process them with the **uncompress** command before you can read them. This is explained in Chapter 18.

Hint

If you FTP to a remote host that has a large number of files, the **ls-lR.Z** and **ls-ltR.Z** files may be so large that you cannot realistically download them. If you do, you may find that the act of downloading will use up so much space on your local computer that your quota is reached and the download is aborted. And, even if you do manage to download such files, they will be even larger when you uncompress them.

Thus, be sure to check the size of any file before you download it. It is not uncommon to have someone use up all the free disk space on the local computer by downloading a huge file. If you find yourself in this position, be courteous and remove the file as quickly as you can. (Use the Unix **rm** command.)

Requesting Anonymous FTP Services by Mail

If you do not have access to an **ftp** program, there is a way that you can request Anonymous FTP services by mail. You use a program called Ftpmail (which was written by Paul Vixie, who was then working at the DEC Network Systems Lab).

Even if you regularly use the **ftp** program, you may occasionally find it convenient to make FTP requests by mail, for instance, when you need a long directory listing. (See the last example in this section.) You may also find that it is sometimes easier to send a request by mail than to use the **ftp** program and do all the work yourself.

To use Ftpmail, you mail a message to an Ftpmail server. There are several Ftpmail servers on the Internet, the principal one being a host named **ftpmail.decwrl.dec.com**. In this section, we will discuss how to use this particular Ftpmail server. At the end of the section, we will show you a list of the other Ftpmail servers that you might use.

To use Ftpmail, you mail a message to **ftpmail@decwrl.dec.com**. Within the message, you enclose commands, one per line. The commands are instructions for conducting an Anonymous FTP session. The Ftpmail server will receive the message and carry out the commands on your behalf. The results of the session

will be mailed to you automatically. In addition, Ftpmail will mail you a note when it receives your request and another note reporting on the actual Anonymous FTP session. (We discuss the Internet mail facility in Chapters 5 and 6.)

When you mail a message to Ftpmail, it ignores anything in the **Subject** line. However, if you do specify a subject, Ftpmail will use it as part of the **Subject** line when it mails you back a note. Thus, if you are mailing more than one request to Ftpmail, it is handy to use different subjects to help you identify the replies.

To get instructions on how to use Ftpmail, send a message to **ftpmail@decwrl.dec.com**. Here is a short session in which we use the **mail** program (Chapter 6) to send such a message:

```
% mail ftpmail@decwrl.dec.com
Subject: Request for help
help
CTRL-D
EOT
```

Before you use Ftpmail, it is a good idea to read all the instructions. For reference, Figure 16-4 contains a summary of the most important Ftpmail commands. In order for these commands to make sense, you should understand the general principles of Anonymous FTP downloading that we explained earlier in the chapter. Note the Ftpmail command to change directories is **chdir**, not **cd**.

Using Ftpmail is straightforward. To make things easy, use the Ftpmail commands in the order that they are listed in Figure 16-4, leaving out the commands that you don't need. For example, to retrieve a text file, you would normally use the following commands (in this order): **reply**, **connect**, **ascii**, **chdir**, **get**, **quit**.

The following example shows a sample session in which we use commands to connect to the Anonymous FTP host **cathouse.org**, change to the **/misc/fun/humor/british.humour/monty.python/flying.circus** directory, and request the text file named **argument.clinic**.

```
% mail ftpmail@decwrl.dec.com
Subject: Example of requesting a text file
reply harly@fuzzball.ucsb.edu
connect cathouse.org
ascii
chdir/misc/fun/humor/british.humour/monty.python/flying.circus
get argument.clinic
quit
CTRL-D
EOT
```

Specifying Your Mail Address	
`reply` *address*	mail response
Connecting to the Host	
`connect` *host*	connect to specified Anonymous FTP host
Setting Options	
`ascii`	files to be mailed are text files
`binary`	files to be mailed are binary files
`uuencode`	convert binary files to text using `uuencode`
`btoa`	convert binary files to text using `btoa`
Specifying the Directory	
`chdir` *directory*	change to the specified directory
Requesting Files	
`get` *file*	send a copy of the specified file
Requesting Information	
`help`	send a description of a how to use Ftpmail
`dir [`*directory*`]`	send a long directory listing
`ls [`*directory*`]`	send a short directory listing (names only)
Ending the Session	
`quit`	terminate session, ignore rest of message

Figure 16-4. *Summary of important Ftpmail commands*

If the file you are retrieving is large (more than 64,000 characters), Ftpmail will automatically split it in to pieces and mail each piece separately. It is up to you to take the separate messages and put them back together to re-create the original file. (Don't worry, they will be numbered.)

Remember that the results of your request will be send to you by mail. And, as we explained in Chapter 5, only text files can be sent by regular mail. Thus, if you request a binary file, it must first be converted to a text file. (Remember that files whose names end with **.Z** or **.tar** are binary files.) At the other end, you will have to decode the text and re-create the original binary file.

There are two common systems that are used to perform such conversions, **uuencode/uudecode** and **btoa** (binary to ASCII). We discuss these programs and how to use them in Chapter 18. For now, we will say that if you ask for a binary

file, you will have to tell Ftpmail how you want it encoded. Simply use either the **uuencode** or **btoa** command. A good place to do this is right after the **binary** command (which tells Ftpmail that you are requesting a binary file). If you send a binary file and do not specify what type of encoding you want, Ftpmail will use **btoa** by default (which may not be what you want).

Thus, to request a binary file, you would normally use the following commands (in this order): **reply**, **connect**, **binary**, **uuencode** (or **btoa**), **chdir**, **get**, **quit**. The next example shows a sample session in which we use commands to connect to the Anonymous FTP host **ftp.uu.net**, change to the **/doc/literary/obi/DEC/humor** directory, and request the text file names **Lawyer.jokes.Z**. (After we receive this file, we will have to use the **uudecode** program to recover the original binary file, and the uncompress programs to uncompress the file. This is all explained in Chapter 18.)

```
% mail ftpmail@decwrl.dec.com
Subject: example of requesting a binary file
reply harley@fuzzball.ucsb.edu
connect ftp.uu.net
binary
uuencode
chdir /doc/literary/obi/DEC/humor
get Lawyer.jokes.Z
quit
CTRL-D
EOT
```

Our last example requests a directory listing by mail. Requests such as this are handy even when you have access to the **ftp** program. In this example, we will request a large list of humor files. Since we will receive this list as a mail message, it is easy to save it for further reference, or even forward a copy to a friend.

The Anonymous FTP host **cathouse.org** contains a wonderful repository of humor called the Catstyle Archives, compiled by Jason R. Heimbaugh (otherwise known as Catstyle). In the following example, we send Ftpmail commands to connect to **cathouse.org**, change to the **/misc/fun/humor** directory, and request a long listing of all the sub-directories (the command **dir** * specifies that we want to look into all the sub-directories). Once we receive this directory listing, we can scan it for interesting files to request.

```
% mail ftpmail@decwrl.dec.com
Subject: Example of requesting a directory listing
reply harley@fuzzball.ucsb.edu
connect cathouse.org
ascii
chdir /misc/fun/humor
dir *
quit
CTRL-D
EOT
```

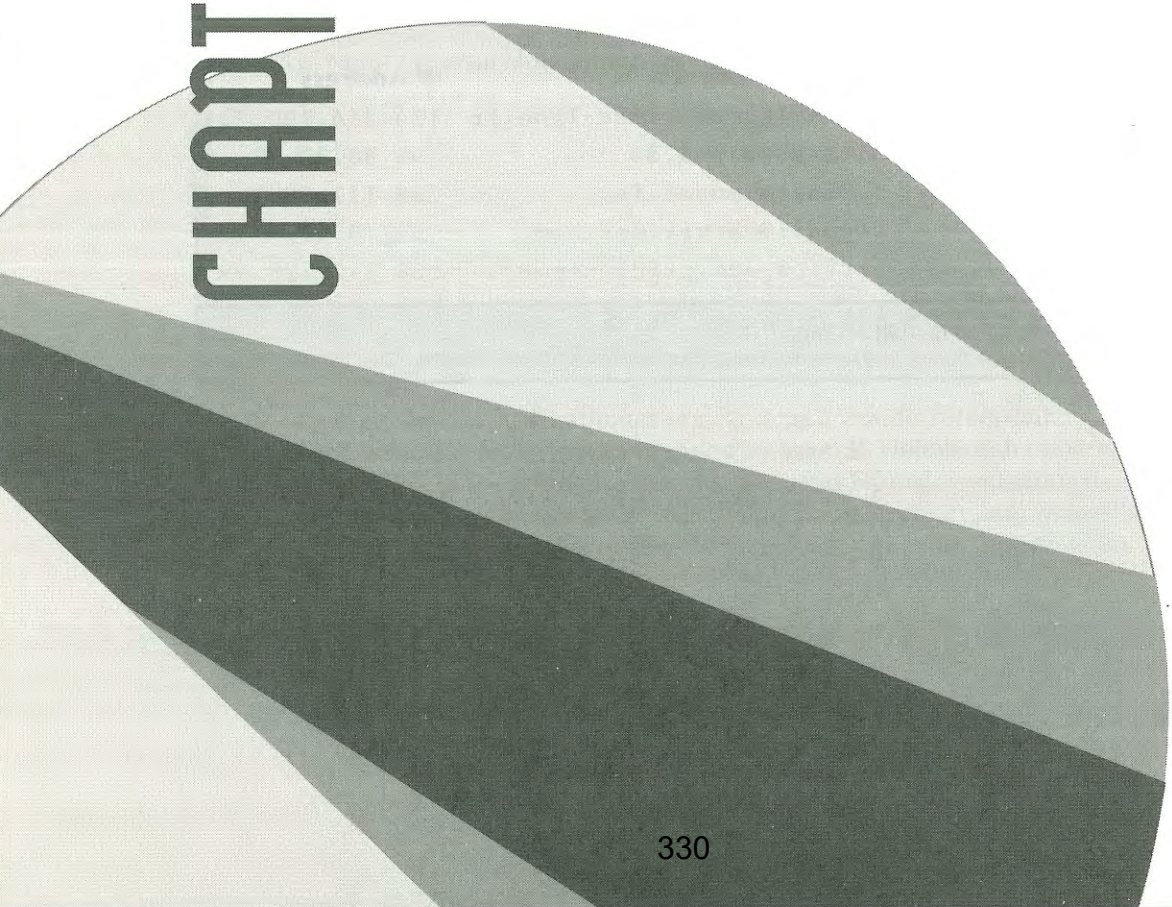
As we mentioned earlier, there are several other Ftpmail servers on the Internet. Table 16-1 shows their addresses. In spirit, they work the same way that we have described in this section, although the details and some of the commands are different.

Before you use one of these Ftpmail servers for the first time, send a message asking for instructions. Simply mail a message that contains only one line with the word **help**, just like we did above.

Location	Internet Address	IP Address
France	ftpmail@grasp.insa-lyon.fr	134.214.100.25
Germany	bitftp@vm.gmd.de	192.88.97.13
Ireland	ftpmail@ieunet.ie	192.111.39.1
USA: California	ftpmail@decwrl.dec.com	16.1.0.1
USA: New Jersey	bitftp@pucc.princeton.edu	128.112.129.99

Table 16-1. *Ftpmail Servers*

CHAPTER 17



Using Archie to Find Anonymous FTP Files

As we explained in Chapter 16, Anonymous FTP provides us with one of the largest collections of public information ever amassed by human beings. Best of all, the Anonymous FTP archives are available for free to anyone on the Internet. Indeed, as we discussed at the end of that chapter, even people who have only Internet mail access can request Anonymous FTP files.

In order to download a file via Anonymous FTP, you need to know where that file is located. In particular, you need to know the address of an Anonymous FTP host and the name of the directory that contains the file. This is not a lot to ask, but, in the vast world of Anonymous FTP, how do you know where to look?

The answer is you use Archie: the card catalog for the largest library in the world.

What Is Archie?

Throughout the Internet, there are a number of computers, called *Archie servers*, that provide a very important service: to help you find the names of Anonymous FTP hosts that carry a particular file or directory. The name "Archie" was chosen because it sounds like the word "archive". Because of its name, we tend to talk about Archie as if it were a person, or at least an intelligent robot.

When you need to find an Anonymous FTP file or directory, all you have to do is tell Archie what you are looking for. Archie will search its database and display the name of each Anonymous FTP host that has that file or directory. Archie will also show you the exact directory path. Thus, all you need to do is FTP to that host and pick up what you want.

There are three ways to use an Archie server, and we will talk about all of them in this chapter. First, you can telnet to it and log in with a userid of **archie**. (We explain how to telnet to a remote host in Chapter 7.) Once you log in, an Archie program will start automatically. You enter commands, one at a time, and tell Archie what you want to look for. Archie will search its database and display the results. If you are not sure exactly what file you want, Archie offers another facility—called a "whatis" service—that has descriptions of thousands of different programs, data files, and documents.

The second way to utilize Archie is by using a program, called an Archie client, that runs on your own computer. (We explain about clients and servers in Chapter 2.) You tell the Archie client what you are looking for. It automatically connects to an Archie server, asks it to perform the search, accepts the output on your behalf, and then displays the results for you. Once you know how to use Archie, it is a lot easier and a lot faster to use an Archie client. You don't have to initiate a telnet session, nor do you have to remember which Archie commands to use.

Finally, you can mail a request to any Archie server. Archie will carry out your request and mail the results back.

How Does Archie Work?

Conceptually, the workings of Archie are surprisingly simple. At regular intervals, special programs connect to every known Anonymous FTP host and download a full directory listing of all the public files. These lists are stored in what is called the *Internet Archives Database*. When you ask Archie to look for a file, all it needs to do is check the database.

The various Archie server sites around the world each keep track of the Anonymous FTP hosts in a certain portion of the Internet. For example, the Australian Archie host keeps track of all the Australian Anonymous FTP hosts. This information is shared so that all Archie servers are kept up to date as much as possible. On the average, Anonymous FTP hosts are checked about once a week.

Of course, there are a lot of details to take care of, so we should be grateful that someone else is providing this service. At the time we wrote this chapter, Archie servers were keeping track of more than two million files on over 1,500 different Anonymous FTP hosts. In total, these files represent hundreds of gigabytes (billions of characters) of data.

Archie was originally developed as a project by students and volunteer staff at the McGill University School of Computer Science in Montreal, Canada. The software was written by Alan Emtage and Bill Heelan with help from Peter Deutsch. Today, Archie is maintained by Bunyip Information Systems in Montreal, Canada. (You can reach them by sending mail to **info@bunyip.com**.)

Getting Started with an Archie Server

To use an Archie server, telnet to it and log in using a userid of **archie**. (You will not need a password.) Table 17-1 contains a list of the public Archie servers available to Internet users. For example, the following command will telnet to the Archie server at Rutgers University:

```
telnet archie.rutgers.edu
```

Hint

You can use any Archie server you want, but you will probably find it faster to use one that is relatively close to you.

Archie servers generally have a limit on how many people can telnet to them at the same time. If the closest Archie server is busy, use the next closest one, or wait for a while and try again.

Once you log in as **archie**, you will see some welcoming messages. You will then see the following prompt:

```
archie>
```

Archie is now ready for your commands. To stop Archie, enter the command **quit**. To display help information, use the **help** command. (We will talk more about **help** later, because there are some nuances.)

Location	Internet Address	IP Address
Austria	archie.edvz.uni-linz.ac.at	140.78.3.8
Austria	archie.univie.ac.at	131.130.1.23
Australia	archie.au	139.130.4.6
Canada	archie.uqam.ca	132.208.250.10
England	archie.doc.ic.ac.uk	146.169.11.3
Finland	archie.funet.fi	128.214.6.102
Germany	archie.th-darmstadt.de	130.83.22.60
Israel	archie.cs.huji.ac.il	132.65.6.15
Japan	archie.wide.ad.jp	133.4.3.6
South Korea	archie.sogang.ac.kr	163.239.1.11
Spain	archie.rediris.es	130.206.1.2
Sweden	archie.luth.se	130.240.18.4
Switzerland	archie.switch.ch	130.59.1.40
Taiwan	archie.ncu.edu.tw	140.115.19.24
USA: Maryland	archie.sura.net	128.167.254.179
USA: Nebraska	archie.unl.edu	129.93.1.14
USA: New Jersey	archie.internic.net	198.49.45.10
USA: New Jersey	archie.rutgers.edu	128.6.18.15
USA: New York	archie.ans.net	147.225.1.10

Table 17-1. *Public Archie Servers*

Using Archie is a three-part process. First, you set things up the way you want, according to your preferences. Then you do the actual work. Then you quit. (Just like life.)

Setting Variables with an Archie Server

To control your working environment, you can change the value of quantities called *variables*. Each variable has a name and a value. You can tell Archie how you want something to work by changing the value of a particular variable. For example, you can specify how you want to sort the output of a search by setting the **sortby** variable. You can examine the current value of all the variables by entering the **show** command:

```
show
```

If you only want to display the setting of a single variable, you can specify that variable. For example:

```
show sortby
```

Figure 17-1 shows typical output from a show command. For now, don't worry about what all the variables mean; we will explain the important ones in a moment.

The best way to begin an Archie session is to start with a **show** command. Take a moment to examine the important variables and see if they are set up to your liking. If not, change them by using the **set** command (described in a moment). Once you have everything just the way you want it, you can enter the commands to tell Archie to search for what you want.

Hint

Before you start an Archie search, make sure that the most important variables are set the way you want. These variables (discussed below) are **maxhits**, **output_format**, **pager**, **search**, **sortby**, and **status**. Pay particular attention to the **search** variable. In addition, if you plan on asking Archie to mail you the results of a search, set the **mailto** variable.

As you can see from Figure 17-1, there are three types of variables: boolean, numeric, and string.

A *boolean variable* is one that acts as an on/off toggle switch. (The word "boolean" is a programming term named after the nineteenth century English mathematician George Boole.) To turn a boolean variable on, use the **set** command. To turn a boolean variable off, use the **unset** command.

Archie has only two boolean variables: **pager** and **status**. The **pager** variable determines how Archie will display its output. When **pager** is set, the results of a search will be sent to a special program, called a paging program, that will display the output one screenful at a time. When **pager** is unset, all the output will be displayed on your screen without stopping. Most of the time, this would cause all but the last part of the information to scroll off the top of your screen, so normally you will want to make sure that **pager** is set. If it is not already set, you can do so yourself by using the command:

```
set pager
```

There are three common paging programs used in Unix: **more**, **pg**, and **less**. Archie servers display their output using **less**, so that is the paging program you

```
# 'autologout' (type numeric) has the value '15'.
# 'compress' (type string) has the value 'none'.
# 'encode' (type string) has the value 'none'.
# 'language' (type string) has the value 'english'.
# 'mailto' (type string) is not set.
# 'maxhits' (type numeric) has the value '100'.
# 'output_format' (type string) has the value 'verbose'.
# 'pager' (type boolean) is set.
# 'search' (type string) has the value 'exact'.
# 'sortby' (type string) has the value 'time'.
# 'status' (type boolean) is set.
# 'tmpdir' (type string) has the value '/tmp'.
# 'term' (type string) has the value 'vt100 24 80'.
# 'max_split_size' (type numeric) has the value '51200'.
# 'server' (type string) has the value 'archie.rutgers.edu'.
```

Figure 17-1. *Typical output from an archie show command*

will have to learn. Like many Unix programs, **less** has enough commands to choke a horsefly. Fortunately, you can probably get by with knowing only two commands: press SPACE to display the next screen of information, and press **q** (quit) to return to Archie. For reference, we have included a quick summary of the most useful **less** commands later in the chapter.



FUN TIP: The **more** *program gets its name from the fact that, at the bottom of each screen of output, the program displays the word "more". The* **less** *program is newer and was written to replace* **more**. *The name* **less** *was chosen as a wry comment, because the* **less** *program actually offers more functionality than the* **more** *program. In other words, "***less** *is more".*

The other boolean variable, **status**, is a lot simpler. As Archie performs a search, it can display a status line at the bottom of the screen. When **status** is set, Archie will display this line. When **status** is unset, Archie will not display the line. Normally, you want **status** to be set. If it is not already set, use:

```
set status
```

The second type of variable is a *numeric variable*. As the name implies, you use the **set** command to give these variables a numeric value. There are only two numeric variables that you need to understand: **autologout** and **maxhits**.

The **autologout** variable controls how long Archie will wait for you to enter a command without logging you out. In our example, **autologout** has a value of 15. This means that if you do not enter a command for 15 minutes, Archie will log you out and terminate your connection. To set **autologout** to another value, use a command like:

```
set autologout 25
```

The permissible range is from 1 to 300 minutes.

When you use Archie to search for a file, you will often find that there are many, perhaps hundreds, of Anonymous FTP hosts that contain the file. To speed things up, you can set **maxhits** to tell Archie the maximum number of items that you want to find. When Archie reaches this number, it will stop searching. For example:

```
set maxhits 10
```

Setting **maxhits** is something you will probably want to do because most Archie servers default to a large number (such as 100), and you really only need one good host name.

Hint

To save time, start searching with **maxhits** set to 10. If the results of your search are not adequate, set **maxhits** higher and try again.

The third type of variable is a *string variable*. The name means that these variables store values consisting of a string of characters. The only string variables you need to know about are **mailto**, **output_format**, **search**, and **sortby**. The most important of these is **search**.

The **mailto** variable is used to store a mailing address. As we will explain later, you can use the **mail** command to tell Archie to mail you the results of a search. If you set the **mailto** variable before you issue the **mail** command, Archie will know where to send the output. Otherwise, you will have to specify your address each time you use the **mail** command.

To set the **mailto** variable, use the **set** command and specify your address. For example:

```
set mailto harley@fuzzball.ucsb.edu
```

The **output_format** variable tells Archie in what format you would like the output. You have three choices: **verbose**, **terse**, and **machine**. To make a choice, use the **set** command with the name of the variable, followed by your choice. Be sure to include the _ (underscore) character in the variable name:

```
set output_format verbose
set output_format terse
set output_format machine
```

Normally, you would use either the **verbose** or **terse** format (which we will show you in a moment). Just experiment and pick the one you like. The **machine** format is designed to be used when you will be mailing the result of a search to yourself, and then using a Unix command or a program to manipulate the raw output. You will see an example of this later in the chapter.

Here is an example of one item of output in each of the three formats. We generated this example by using Archie to search for a file named **shoo-fly-pie**. The **verbose** output looked like this:

```
Host mthvax.cs.miami.edu    (129.171.32.5)
Last updated 09:32 17 Jun 1993

    Location: /recipes/ovo
      FILE    -rw-r--r-- 1095 bytes 01:00  4 Dec 1991  shoo-fly-pie
```

Here is the **terse** output:

```
mthvax.cs.miami.edu  01:00  4 Dec 1991 1095 bytes /recipes/ovo/shoo-fly-pie
```

Finally, the **machine** output:

```
19920103010000Z mthvax.cs.miami.edu 1095 bytes -rw-r--r-- /recipes/lacto/shoo-fly-pie
```

The next variable, **search**, is used to tell Archie how you want it to compare patterns as it searches. We will discuss this variable on its own in the next section.

The **sortby** variable indicates in what order you want Archie to display the results of a search. There are several choices:

`set sortby none`	do not sort
`set sortby filename`	alphabetical by file name
`set sortby hostname`	alphabetical by host name
`set sortby size`	largest to smallest
`set sortby time`	newest to oldest

You can tell Archie to sort in reverse order by placing an r in front of the variable setting:

`set sortby rfilename`	reverse alphabetical by file name
`set sortby rhostname`	reverse alphabetical by host name
`set sortby rsize`	smallest to largest
`set sortby rtime`	oldest to newest

For example, each file and directory has a time and date that marks when it was last modified. To tell Archie to display its output sorted by time, with the most recently modified items first, enter:

```
set sortby time
```

Setting the search Variable

The main function that Archie performs is to search the Internet Archives Database for a pattern that you specify. To tell Archie to start the search, you use the **find** or **prog** command (discussed later in the chapter). Before you start the search, you can set the **search** variable to control how Archie looks for a match.

There are seven possible settings: **exact**, **sub**, **subcase**, **regex**, **exact_sub**, **exact_subcase**, and **exact_regex**. To set the **search** variable, use the **set** command, followed by **search**, followed by your choice. For example:

```
set search exact
```

The **exact** setting performs a basic, straightforward search. It tells Archie to look for names that are exactly like the one you specified, including upper- and lowercase. For example, if you asked Archie to search for the pattern **IBM-PC**, it would match **IBM-PC**, but not **IBM-pc** or **ibm-pc**. When you know exactly what you want, these types of searches are the fastest and yield the best results.

The **sub** setting tells Archie to search for patterns that contain your specification as a substring. For example, if you asked Archie to search for **PC**, it would match **IBM-PC**, **PC**, or **PC-dos**. This setting does not distinguish between upper- and lowercase, so **PC** will also match **IBM-pc**, **pc**, and **pc-dos**.

The **subcase** setting is the same as **sub** except that upper- and lowercase letters are considered to be different. With this setting, **PC** would match **IBM-PC**, but not **IBM-pc**.

The **regex** setting allows you to use what is called a *regular expression* to tell Archie what you want. Regular expressions are used within Unix as a compact way to specify general patterns. For example, if you tell Archie to search for the regular expression **PC$**, it means to find all names that end with the letters **PC**.

Regular expressions can be complex, and we won't go into all the details here. If you want more information, use the command **help set search**. (We explain the **help** command later in the chapter.)

The last three settings are **exact_sub, exact_subcase**, and **exact_regex**. They tell Archie to first try an **exact** search. Then, if no matches are found, to try again using the second setting. For example:

```
set search exact_sub
```

This tells Archie to start by searching for the specified pattern exactly. If that fails, then try searching for the pattern as a substring.

Performing a Search

To prepare for an Archie search, use the **show** command, check the settings, and change the ones you don't like. If you are not sure what to set, use the following:

```
set mailto your-mail-address
set maxhits 10
set output_format verbose
set pagerset search exact_sub
set sortby time
set status
```

To perform the search, use the **find** command. Enter **find** followed by the pattern you want Archie to locate. Here are some examples:

```
find shoo-fly-pie
find rfc1325.txt
find recipes.tar.Z
```

Another command that you may need to know about is **prog**, which is a synonym for **find**. When Archie was first invented, its database was used primarily to hold information about computer programs. For this reason, the command that started a search was named **prog**.

Today, Archie's database contains information about all kinds of Anonymous FTP resources: not only programs, but documents, electronic magazines, Usenet archives—just about every type of information that you can imagine. Thus, the search command has been renamed from **prog** to **find**.

However, if you are using an Archie server that uses the old version of the Archie software, there may not be a **find** command. In such cases, you will have to use **prog** instead.

> **Hint**
>
> To stop an Archie search before it finishes, you can press CTRL-C.

Mailing the Results of an Archie Search

Once Archie has completed a search, you may find it useful to mail the results to yourself (or to someone else). To do so, all you have to do is enter the **mail** command:

```
mail
```

If you have set the **mailto** variable, Archie will already know where to send the mail. Otherwise, you will have to specify the address as part of the **mail** command:

```
mail harley@fuzzball.ucsb.edu
```

> **Hint**
>
> If you plan on mailing the results of more than one Archie search, it is a lot easier to set the **mailto** variable once, before you start.

Commands to Use While Reading Archie Server Output

We explained earlier that when you set the **pager** variable, it tells Archie to display its output by using a paging program called **less**. When **less** displays output, it shows you one screenful at a time. After each screenful, **less** pauses and waits for a command.

At this point, you can press SPACE to display more data, or **q** to quit **less** and return to Archie. For reference, Table 17-2 contains commands that you can use as you are viewing Archie output. For most of the commands, just press the keys, you do not press RETURN. However, for the / and ? commands, you do have to press RETURN.

Command	Description
SPACE	go forward one screenful
q	quit the program
RETURN	go forward one line
*n*RETURN	go forward *n* lines
b	go backward one screenful
y	go backward one line
*n*y	go backward *n* lines
d	go forward (down) a half screenful
u	go backward (up) a half screenful
g	go to the first line
*n*g	go to line *n*
G	go to the last line
p	go to the line that is *n*% through the output
/*pattern*	search forward for the specified pattern
?*pattern*	search backward for the specified pattern
n	repeat the previous search command

Table 17-2. *Commands to Use When Viewing Output with* **less**

Using the Whatis Database

If you know the name of a file or directory (or part of the name), you can use the **find** command to have Archie search the Internet Archives Database. However, what if you know what you want, but you don't know the name?

To help in such cases, Archie maintains a second collection of information called the *Software Description Database*. This database contains short descriptions of thousands of programs, documents, and data files that are found in the Anonymous FTP archives (a lot more than software, actually).

To search this database, you use the **whatis** command. Type the command followed by any word you want. Archie will search the Software Description Database and display all the entries it finds that contain the word you specified. Each entry will show you a description and a file name. If the description looks like what you want, you will know exactly which file to search for.

Where do the entries come from? Whenever someone makes a file available via Anonymous FTP, he or she can send a short description to the people who maintain Archie, who will place the description in the Software Description Database.

Thus, there is a built-in restriction that limits the effectiveness of the **whatis** command. If the person who created the file you are looking for did not submit a description—which is the case for most files—you will not find it in the Software Description Database. Moreover, unlike the Internet Archives Database, this one is not updated regularly. Thus, you may find the description of a file that does not exist anymore.

Still, when it can find what you want, the **whatis** command can be a great tool, saving you from long searches in the dark. (The name, by the way, comes from the Unix **whatis** command, which performs a similar function for the online Unix manual.)

Here is an example using **whatis**. You would like to find something interesting that has to do with telephones. Enter:

```
whatis phone
```

Here is part of the output:

```
dialup      Maintain a database of phone services and use cu(1)
            to call them
phone       Multi-user real-time "talk" program
phone_kl    Phone another user, typing screen to screen
phoneme     Translate English words into their phonetic spellings
ringback    Implements a ring-back system that allows a phone
```

```
                line that is normally used as a voice line and a
                dial-out data line to be used as a limited dial-in
                data and voice line
sys5-phone      VAX-like Phone Utility for SysV
telewords       Telephone number to word conversion
telno           A telephone number permutation program
```

You decide to try the **telewords** program. All you have to do is enter the command **find telewords** to display its Anonymous FTP location.

There are several things we would like to point out about this example. First, notice that **whatis** does a straight letter-by-letter search, ignoring the difference between upper- and lowercase letters. Second, you will often find items that have nothing to do with what you are looking for. For example, the **phoneme** program (although it does look interesting).

Finally, remember that **whatis** does not perform an exhaustive search of the full Internet Archives Database. It only knows about the items listed in the Software Description Database. For example, several Anonymous FTP hosts have directories named **telephone** that contain useful information, but **whatis** cannot find these directories because they are not in its database.

Displaying Help Information

There are several ways to have Archie display help information. First, you can use the **help** command to display information about other commands. All you need to do is enter **help** followed by the name of the command. For example:

```
help find
help set
```

To display a list of all the commands, enter:

```
help ?
```

If the **pager** variable is set, Archie will use the **less** paging program to display the information, so when you are finished you will have to press **q**. Once you quit **less**, you will see a new prompt:

```
help>
```

At this point, you can enter the name of one command after another and receive more help. When you do not need any more help, simply press RETURN and you will return to the regular prompt:

```
archie>
```

You can now enter a regular Archie command.

If you want help for setting a variable, type **help set** followed by the name of the variable. For example:

```
help set search
help set pager
```

Again, the information will be displayed using **less**. However, this time, when you quit, you will see the prompt:

```
help set>
```

You can now enter the name of one variable after another and receive more help. When you press RETURN, you will see the **help>** prompt. Pressing RETURN once again returns you to the regular **archie>** prompt.

Hint

In order to go back to the **archie>** prompt after reading **help** information, you will have to press **q** (to quit **less**) and then RETURN (perhaps twice).

Another way to learn about Archie is to read the official documentation. Here is how the Archie documentation is organized.

All Unix systems come with an online manual that contains entries for each command. At any time, you can read the documentation for a particular command by using the **man** command. It is customary to refer to each particular entry in the online manual as a *page* (even though many entries are much longer than a single page). It is common for people who create new software to write a manual page explaining how the software works.

Archie itself has such documentation that explains how to use an Archie server. This documentation, of course, is not available on your local computer. Rather, you can enter the **manpage** command while you are using the Archie server:

```
manpage
```

Hint

Archie will now display its manual page for you.

After you use a **help** or **manpage** command, you can mail the output to yourself by using the **mail** command. This is a good way to get your own copy of the official Archie server documentation.

To finish this section, we will mention three more information commands.

Whenever you log in to an Archie server, you will see a welcoming message. Indeed, many public Internet hosts, such as Anonymous FTP hosts, display general information when you log in. The custom is to call such information the *message of the day*. With an Archie server, you can redisplay this information at any time by entering the **motd** command.

If you would like to find out about other Archie servers, you can display a list by entering the **servers** command.

And finally, you can use the **version** command to display the version of Archie software that your server is using.

A Summary of Archie Server Commands

For reference, Figure 17-2 is a summary of all the important Archie server commands.

Stopping the Archie Session	
`quit`	stop the Archie session and disconnect
Performing a Search	
`find` *pattern*	search the main Anonymous FTP database
`prog` *pattern*	the old name for `find`
`whatis` *pattern*	search the Software Description Database
Displaying Information	
`help ?`	display a list of commands
`help` *command*	display help for the specified command
`help set` *variable*	display help for the specified variable
`manpage`	display the Archie manual page
`motd`	redisplay message of the day
`servers`	display a list of Archie servers
`version`	show what version of Archie is being used
Mailing Information	
`set mailto` *address*	specify a mail address
`mail` [*address*]	mail output
Displaying Variable Settings	
`show`	display the value of all the variables
`show` *variable*	display the value of a specified variable

Figure 17-2. *Summary of important Archie server commands* (continued next page)

Setting General Variables

`set autologout` *minutes*	set maximum idle time before auto logout
`set maxhits` *number*	set the maximum number of items to find
`set pager`	display output by using paging program
`unset pager`	do not display output with paging program
`set status`	display a status line during search
`unset status`	do not display status line during search

Setting Output Preferences

`set output_format verbose`	display output using long format
`set output_format terse`	display output using short format
`set output_format machine`	display output using machine format
`set sortby none`	do not sort output
`set sortby filename`	sort alphabetically by file name
`set sortby hostname`	sort alphabetically by host name
`set sortby size`	sort by size, largest to smallest
`set sortby time`	sort by time and date, newest to oldest
`set sortby rfilename`	sort reverse alphabetically by file name
`set sortby rhostname`	sort reverse alphabetically by host name
`set sortby rsize`	sort by size, smallest to largest
`set sortby rtime`	sort by time and date, oldest to newest

Setting Search Preferences

`set search exact`	search for an exact pattern
`set search sub`	search for anything that contains pattern
`set search subcase`	same as `sub`, but distinguish case
`set search regex`	search for a regular expression
`set search exact_sub`	try `exact`, then `sub`
`set search exact_subcase`	try `exact`, then `subcase`
`set search exact_regex`	try `exact`, then `regex`

Figure 17-2. *Summary of important Archie server commands* (continued)

A Typical Session with Archie

It was a dark and stormy night.

Outside, the harsh, driving rain beat a monotonous tattoo against the windowpane. We sat at our desk, hat brim pulled down over our face, grabbing a few quick winks between cases. Things had looked tight for a while in Tangier, but once we had put the screws to the dwarf he had come through. They all do.

We lay back, the old number twelves resting on a desk covered with a month's worth of paperwork, a half-empty bottle of Scotch, and a tarnished .45 that was still smoking. In the corner, the screen of our computer glowed silently.

We didn't hear when she entered, but our nostrils flared at the fragrance of her perfume. "Go away," we said, "we're closed for repairs."

"Well, then," said a sultry Barbara Stanwyck voice, "you'll just have to open early today. I need help, and I'm willing to pay for it."

We looked up and blinked twice at a vision of innocence: five foot three, periwinkle blue eyes, and long blond curls winding over her shoulders like water cascading down a mountain creek. And she was dressed in a skintight outfit that showed more curves than a politician with the bends.

She looked innocent. Too innocent. Something was fishy here, and we didn't mean her perfume.

"Listen," she said, "I need help, and you're the only one who can help me. I'm in an awful jam."

"So what else is new?"

"I've only got seven more hours, or else..."

"Or else what? We already told you, kid, that we weren't going to pull your chestnuts out of the fire anymore. What is it this time? Blackmail? Murder? Or did you have another misunderstanding with Sergeant Rogers?"

She looked at us with large, round, innocent eyes.

"Calculus," she said.

We closed our eyes and sighed.

"Okay, kid, spill it."

Well, we listened to her tale and it wasn't a pretty one. The old, old story. A young, guileless girl from a small country town, still wet behind the ears, full of hope and dreams of tinsel, comes to the big city to make her fortune. But then she falls in with the wrong crowd and gets into trouble.

It starts innocently enough. First, someone takes her to a party where, unknown to her, people in the back room are solving differential equations. Next, she gets invited to a small, private gathering where a few street-wise punks introduce her to basic algebra. At first, the thrill is intoxicating and she goes along with it just for the kicks. Soon, she's into simultaneous equations, conic sections, and fractional exponents. She tells herself that she can take it or leave it, that she can stop whenever she wants. Until one day, she finds herself breaking into a pawn shop to steal a protractor and a Schaums Outline.

And now, calculus.

"You have to help me," she pleaded. "You're my last hope. My first calculus class starts tomorrow and I'm not ready. And you know what The Doctor does to people who are unprepared."

We sighed again.

"Okay," we said, "we'll help you, but it will cost you a hundred bucks. In advance."

"How about a trade," she said. "I know where I can get my hands on a used copy of a complex analysis text."

"That's all we need around here, with the cops breathing down our neck. A hundred bucks, in advance. Think on your feet."

"All right," she pouted and, reaching into the hidden recesses of her outfit, extracted two fifty-dollar bills that had definitely seen better days.

"That's more like it," we said. "Now come over here and sit down."

We pulled her over to the computer which was already logged in to the Internet.

"Do you have your own PC at home?"

She nodded.

"Good. We'll get Archie to find a program to help you."

Telneting to a handy Archie server, it was the work of a moment to set the variables and start a search for anything named "calculus". We hit paydirt on the first try: an Anonymous FTP host at Washington University, in St. Louis, Missouri. (The session is shown in Figure 17-3.) We quit Archie, FTP-ed to the Anonymous FTP host, changed to the **calculus** directory, and displayed a directory listing.

"There it is," we said. "This is what you want. A file named **rurc1.zip**. Do you know how to uncompress a zip file?"

She looked at us with scorn. "Does your grandmother know how to suck eggs?"

"Okay, then, all you have to do is unzip this file and you will find a program called *Are You Ready for Calculus?* written by David Lovelock at the University of Arizona. It will help you figure out what you need to review before you start calculus."

She stared at us.

"How do you know all about this?" she asked suspiciously.

We stared back.

"That's what you're paying us for, kid."

"Okay. I guess I know better than to ask questions."

We downloaded the program to our computer and copied it onto a floppy disk for her. She took the disk, slipped it into her blouse, and walked to the door.

She turned and looked at us with those large, blue, innocent eyes.

"I like your style," she said. "Maybe you'd like to come up to my place and uncompress a file sometime." She paused. "You do know how to uncompress, don't you? You just put your fingers on the keyboard—and unzip."

And she was gone, like a wisp of steam on a double-sided razor blade ready to do its work.

We looked outside. It was still a dark and stormy night.

```
% telnet archie.rutgers.edu
Trying 128.6.18.15 ...Connected to dorm.Rutgers.EDU.
Escape character is '^]'.

SunOS UNIX (dorm.rutgers.edu) (ttyq0)
login: archie

Last login: Wed Jun 23 16:33:38 from fuzzball.ucsb.edu
SunOS Release 4.1.3 (TDSERVER-SUN4C-DORM) #1:
Sat May 1 16:46:07 EDT 1993

          Welcome to the Rutgers University Archie Server!

...message of the day deleted...

archie> show
# 'autologout' (type numeric) has the value '15'.
# 'compress' (type string) has the value 'none'.
# 'encode' (type string) has the value 'none'.
# 'language' (type string) has the value 'english'.
# 'mailto' (type string) is not set.
# 'maxhits' (type numeric) has the value '100'.
# 'output_format' (type string) has the value 'verbose'.
# 'pager' (type boolean) is not set.
# 'search' (type string) has the value 'sub'.
# 'sortby' (type string) has the value 'none'.
# 'status' (type boolean) is set.
# 'tmpdir' (type string) has the value '/tmp'.
# 'term' (type string) has the value 'vt100 24 80'.
# 'max_split_size' (type numeric) has the value '51200'.
# 'server' (type string) has the value 'archie.rutgers.edu'.

archie> set maxhits 10
archie> set pager
archie> set search exact_sub
archie> set sortby time

archie> find calculus

...some responses deleted...

Host wuarchive.wustl.edu   (128.252.135.4)
Last updated 05:03 23 May 1993

    Location: /edu/math/msdos
      DIRECTORY rwxr-xr-x      1024  Mar  9 18:02   calculus
```

Figure 17-3. *A typical Archie and Anonymous FTP session* (continued next page)

```
archie> quit
# Bye.
Connection closed by foreign host.

% ftp wuarchive.wustl.edu
Connected to wuarchive.wustl.edu.
220 wuarchive.wustl.edu FTP server
(Version 2.1WU(2) Wed May 19 07:29:30 CDT 1993) ready.

Name (wuarchive.wustl.edu:harley): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:

...message of the day deleted...

230 Guest login ok, access restrictions apply.

ftp> cd /edu/math/msdos/calculus
250 CWD command successful.

ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 2344

...some lines deleted...

-rw-r--r--   1 husch    234        171950 Jan 10 18:07 rurc1.zip
-rw-r--r--   1 husch    234         99621 Jan 10 18:08 rurc2.zip
-rw-r--r--   1 husch    234        108018 Jan 11 07:47 rurc3.zip

226 Transfer complete.
2657 bytes received in 5.8 seconds (0.45 Kbytes/s)

ftp> get rurc1.zip
200 PORT command successful.
150 Opening ASCII mode data connection for rurc1.zip (171950 bytes).
226 Transfer complete.
local: rurc1.zip remote: rurc1.zip
172885 bytes received in 1.4e+02 seconds (1.2 Kbytes/s)

ftp> quit
221 Goodbye.
```

Figure 17-3. *A typical Archie and Anonymous FTP session (continued)*

Understanding Archie Clients

Up to now, we have been talking about using an Archie server directly, by telneting to it, logging in as **archie**, and entering commands. However, once you understand how Archie works, the preferred way to use it is via an Archie client.

An *Archie client* is a program that you run on your own computer. You tell the program what you want to search for. It contacts an Archie server on your behalf, issues the request, accepts the output, and displays it on your screen.

Using an Archie client is not only easier, it can be astonishingly fast. We have often found that with an Archie client, we can get a response in only a few seconds. (Of course, when the server gets busy, your response time can slow down considerably.)

Another consideration is that most Archie servers enforce strict quotas about how many people can telnet in at the same time, and you may not be able to connect whenever you want. Archie clients, being programs that run on your own computer, are always available.

There are various types of Archie clients available. The two most common are **archie** (for regular Unix terminals) and **xarchie** (for X Window terminals, see Chapter 2). There also Archie clients for Next computers, PCs running DOS, and VAX computers running VMS.

The **xarchie** program uses an X Window based graphical user interface and, if you understand how to use Archie, is so easy to use that you won't need our help. So, in this section, we will concentrate on **archie**.

To use **archie**, it must be installed on your system. To find out if this is the case, try performing a search by entering the **archie** command followed by something to search for. For example:

```
archie rfc1325.txt
```

If you have the **archie** program, this command will return some type of response. Otherwise, you will see a message like:

```
archie: Command not found.
```

If this is the case, the only ways for you to access Archie are to either telnet to an Archie server, or to send a request by mail (explained later in the chapter). Since using an Archie client is so much more convenient, you may want to politely ask your system manager if he or she could install **archie** on your system. The program is available by Anonymous FTP. (Just use Archie to search for **archie**.)

If you have the **archie** program on your system, you can read the official documentation from the online manual by using the **man** command:

```
man archie
```

Hint

As you read about Archie, you may see references to Prospero. This is a tool that allows you to access information spread around the Internet. Using Prospero, you can organize and use files that are stored on remote computers as easily as if they were on your own local computer.

The distributed file system that Prospero creates is based on an idea called the Virtual System Model, and is widely used throughout the Internet. Anyone can use Prospero by installing a Prospero client on their computer. Archie clients and servers use the Prospero technology to implement their client/server relationship, although the details are hidden from view.

Prospero was developed by Cliff Neuman of the Information Sciences Institute, a part of the University of Southern California. For more information, you can FTP to **prospero.isi.edu**. You can also subscribe to a general Prospero mailing list by sending a message to **prospero-request@isi.edu**. (We discuss how to use FTP in Chapter 16; mailing lists in Chapter 25.)

Using an Archie Client

In order to learn how to use an Archie client program, you should already understand about Archie and how it works. If you need a refresher, read through the previous sections in this chapter.

The easiest way to use **archie** is to specify the name of a file or directory that you want to find. For example:

```
archie rfc1325.txt
```

The Archie client will connect to an Archie server, have it perform a search, and return the results to you.

One nice thing about using an Archie client is that you can manipulate the output using the standard Unix tools. For example, the following command performs the same Archie search and saves the output in a file named **rfc**:

```
archie rfc1325.txt > rfc
```

If a file named **rfc** does not already exist, Unix will create it for you. If the file does exist, it will be replaced and the original contents will be lost, so be careful.

Hint

In Unix, you can use a > (greater than) character followed by a file name at the end of any command to send output to the specified file. This is called "redirecting the standard output". To find out more about such matters, we refer you to any good Unix book. (The best one is *A Student's Guide to Unix*, by Harley Hahn, published by McGraw-Hill.)

As with many Unix commands, **archie** has a number of ways for you to change the behavior of the command. To do so, you include what are called *options* or *switches* directly after the command name. For example, the **-t** option tells Archie to sort the output by the time and date that the file was last modified, from newest to oldest:

```
archie -t rfc1325.txt
```

This is like using **set sortby time** with an Archie server.

Table 17-3 contains a summary of the most important options. Notice that many of them correspond to Archie server commands.

The **-e**, **-s**, **-c**, and **-r** options have the same effect as using the **set search** command with an Archie server. For example, to perform a **sub** search you can use the **-s** option:

```
archie -s rfc1325
```

The **-e** option performs an **exact** search, which is the default. Thus, the following two commands are equivalent:

```
archie rfc1325.txt
archie -e rfc1325.txt
```

If you want to perform a combination search, such as **exact_sub**, you can use **-e** with another search option. For example:

```
archie -e -s rfc1325
```

This command tells Archie to start with an **exact** search. If that doesn't work, Archie will try a **sub** search.

Similarly, you can use the **-t** and **-l** options to control the formatting of the output. The **-t** option sorts by time (like **set sortby time**), and the **-l** option formats the output so that it is suitable to send to another program or Unix command for further processing (like **set output_format machine**):

```
archie -l rfc1325.txt
```

Option	Description
`-c`	search for substrings, case sensitive (`set search subcase`)
`-e`	search for an exact match [default] (`set search exact`)
`-r`	search using a regular expression (`set search regex`)
`-s`	search for substrings (`set search sub`)
`-o`*filename*	send output to specified file
`-l`	list one item per line (`set output_format machine`)
`-t`	sort output by time and date (`set sortby time`)
`-m`*number*	set maximum number of items to find (`set maxhits`)
`-h`*address*	send Archie server requests to the specified host
`-L`	show list of Archie servers known to the program (`servers`)
`-V`	(verbose) make comments during a long search

Table 17-3. *Summary of Important* **archie** *Options*

Here is some sample output from the previous command. Notice that Archie has formatted the output so that each item is on a single line:

```
19920515000000Z  91885 esel.cosy.sbg.ac.at /pub/mirror/rfc/rfc1325.txt
19920515000000Z  91885 swdsrv.edvz.univie.ac.at /doc/rfc/rfc1325.txt
19920514000000Z  91884 plaza.aarnet.edu.au /rfc/rfc1325.txt
19920601000000Z  91884 sunb.ocs.mq.edu.au /Documents/RFC/rfc1325.txt
19920817000000Z  91885 sifon.cc.mcgill.ca /pub/ftp_inc/doc/rfc/rfc1325.txt
```

If you know how to use Unix, you can send this type of output to another command. For instance, the following series of commands displays all the items that are stored on Australian computers (those whose addresses end in **.au**):

```
archie -l rfc1325.txt | grep '.au '
```

For the details of how the **grep** command works, check with a good Unix book. (Guess which one we recommend?)

The **-m** option has the same effect as the Archie server command **set maxhits**. For example, to ask for a maximum of 10 items, use a command like the following:

```
archie -m10 rfc1325.txt
```

Hint

To speed up an Archie search, use the **-m** option to limit the amount of output. (We recommend **-m10**.) If you do not find what you want, re-enter the command with a larger value.

As we showed above, you can use the Unix redirection facility to send the output of an Archie search to a file. Alternatively, you can do the same thing with the **-o** option. Thus, the following two commands are equivalent:

```
archie rfc1325.txt -orfc
archie rfc1325.txt > rfc
```

When you use the **archie** program, it connects to the Archie server whose name was specified by your system manager when he or she installed the program. If you want to see the name of this server, enter the following:

```
archie -L
```

This command will also display a list of other Archie servers. If you would like to send a request to a specific Archie server, you can use the **-h** (host) command and specify the server you want. For example, the following command sends a request to the Archie server whose address is **archie.au**:

```
archie -harchie.au rfc1325.txt
```

Finally, the last option that we will discuss is useful when you are requesting a time-consuming search. By using **-V** (verbose), you tell your Archie client to make comments every now and then:

```
archie -V rfc1325.txt
```

Whenever the search seems to be taking a long time, you will see a suitable message:

```
Searching...
```

In this way, you are reassured that something is happening, and your level of anxiety remains at a comfortable level.

Using Archie by Mail

We have already discussed two ways to use Archie. The best way is to use an Archie client program (like **archie** or **xarchie**) that will send requests to an Archie server on your behalf. If this is not possible, you can telnet to an Archie server, log in as **archie**, and enter commands for yourself. In this section, we will describe the third way to access an Archie server: by sending it commands in a mail message.

Using an Archie server by mail is handy if you do not have access to a client and, for some reason, you cannot telnet directly to an Archie server. It is also convenient when you do not need your answer in a hurry or when you need to make multiple searches. You can mail your request and pick up the results later in your mailbox.

To use Archie by mail, send a message to the userid **archie** at the address of one of the Archie servers (which are listed in Table 17-1). When you compose the

message, you can leave the subject blank. Within the body of the message, use as many Archie commands as you want, each one on its own line.

Archie will carry out your commands and mail a response back to you. For the most part, the commands that you can use are the same ones that we described earlier in the discussion of Archie servers. (See the summary in Figure 17-2.) The only exception is that you cannot use commands that do not make sense in a mail request (for example, **set pager**).

The first command that you should always use is **set mailto**. This will ensure that Archie has your correct return address. If you leave out this command, Archie will look for your address in the header of your message. Most of the time, this will work, but it is better to specify the address explicitly, so you know that there will not be any ranygazoo. For convenience, you can also use the older version of this command, which is **path**. Thus, the following two commands are equivalent:

```
set mailto harley@fuzzball.ucsb.edu
path harley@fuzzball.ucsb.edu
```

The last command that you should always use is **quit**. This tells Archie that you are finished and to ignore any lines that follow. It is important to include the **quit** command, because some mailing programs allow you to define a signature that will be appended automatically to all your messages. By using **quit**, you ensure that Archie will ignore any extra lines at the end of the message.

Before you use a particular Archie server by mail for the first time, you should send it a message asking for help information. This will show you exactly what commands that particular Archie server will recognize.

Here is a sample session in which we send such a request to the Archie server named **archie.rutgers.edu**:

```
% mail archie@archie.rutgers.edu
Subject:
set mailto harley@fuzzball.ucsb.edu
help
quit
CTRL-D
EOT
```

When you get your response, read it carefully. If your Archie server is using an old version of the Archie software, you may not be able to use all the commands in Figure 17-2. In particular, instead of **set mailto**, you will have to use **path**; and instead of **find**, you will have to use **prog**.

Here is a sample session in which we compose a request for Archie to search for the notorious **rfc1325.txt** file:

```
% mail archie@archie.rutgers.edu
Subject:
set mailto harley@fuzzball.ucsb.edu
set maxhits 25
set output_format verbose
set search exact
set sortby time
find rfc1325.txt
quit
CTRL-D
EOT
```

Once you receive your response from Archie, you can send a message to an Anonymous FTP mail server to retrieve the actual file. (See Chapter 16.)

CHAPTER 18



File Types Used on the Internet

In this chapter, we will explain what file types you can expect to encounter on Anonymous FTP servers and within Usenet newsgroups. You will find this information essential if you want to access many of the Internet's public files.

We will also answer two questions that are invariably asked by new users: How do I copy a file to or from my personal computer? And, how do I display picture files?

363

Using Archie to Find File Utility Programs

Before we get into the main topics of this chapter, here is a quick prelude.

Within this chapter, we will be using a number of file utility programs. Some of the names are **compress**, **uncompress**, **tar**, **uuencode**, **uudecode**, **btoa**, and **shar**. In our examples, we assume that you are using a Unix computer, in which case, most or all of these programs should be available on your system.

If you are using another type of computer—such as a PC or Macintosh—there are public domain versions of these programs available via Anonymous FTP.

There are two easy ways to find these programs. First, you can check the file utility reference that is maintained as a public service by David Lemson at the University of Illinois. To download this document, FTP to **ftp.cso.uiuc.edu** and look in the directory **/doc/pcnet** for a file named **compression**.

The second way to find a file utility program is to use Archie (see Chapter 17). The easiest method is to use a **regex** search, and look for the name of the program you want. To save time, specify a search pattern that starts with a ^ (circumflex) character. This means that the pattern must come at the beginning of a word. For example, you might search for **^tar**. This would match **tar.exe** but not **game.tar.Z**.

Here are three examples that search for the **tar** program. First, using an Archie client:

```
archie -m25 -r ^tar
```

You will usually find it handy to send the output of an Archie search to a paging program, in order to display the output one screenful at a time:

```
archie -m25 -r ^tar | more
```

Second, using Archie by mail, send a message that contains commands similar to the following ones:

```
set mailto harley@fuzzball.ucsb.edu
set maxhits 25
set sortby time
set search regex
find ^tar
quit
```

Third, after telneting to an Archie server and logging in as **archie**, enter the following commands:

```
set maxhits 25
set pager
set sortby time
set search regex
find ^tar
```

For more details on using Archie, see Chapter 17.

File Types Used with Anonymous FTP

The people who set up Anonymous FTP servers have two important file storage problems to consider. First, many files are large and require a great deal of disk space. Large files also take longer to download than small files. Second, many of the items offered to the public are actually packages and not single files. A single piece of software, for example, might consist of a number of separate programs and documentation files. In such cases, the package itself needs to be easy to store and to download.

For these reasons, Anonymous FTP files are commonly stored in two special kinds of formats. First, large files are usually *compressed* into a form that requires less storage space. After you download such a file, you must use a program to *uncompress* the file before you can use it.

Second, a group of files can be collected into a single package called an *archive*. The archive can be stored and downloaded as a single file. This is analogous to how you might move books from one place to another. You can place a collection of books into a single box, which is easy to store and easy to move. Once the box has been moved, you can open it and take out the books.

Similarly, you can download an archive as a single large file. Once the archive is on your local computer, you use a program to restore the original collection of files. When you do, we say that you are *unpacking* the archive. We also say that you are *extracting* the files.

There are a number of programs that are used to compress and archive files. The most common compression program is named **compress**. When you use this program, it creates a new file that has the same name as the old one with the characters **.Z** added. For example, when you compress a file named **recipe**, it generates a new (smaller) file named **recipe.Z**.

Thus, whenever you see a file whose name ends in **.Z**, you know that this is a compressed file that will have to be uncompressed before you can use it. To do so, you use a program named **uncompress**.

When it comes to archiving, the most commonly used program is **tar**. (The name stands for "tape archiver", from the days when files were archived and stored on magnetic tape.) When you use **tar**, it creates a new file whose name ends in **.tar**. For example, you might archive a group of files that contains all the programs and documentation for a software package named **game** into an archive named **game.tar**. Such archives are commonly called *tarfiles*.

Thus, whenever you see a file whose name ends in **.tar**, you know that this is an archive that must be unpacked before you can use the files. To do so, you use the **tar** program on your local computer.

Hint

A designation like **.Z** or **.tar** that comes at the end of a file name is called an *extension*. Within the world of Unix and the Internet, there are a number of different extensions that are used to indicate file types.

If you have ever used a PC with DOS, you will be used to extensions that must be from one to three letters. In Unix, the rules are more liberal. Extensions are considered to be part of the file name and can be as long as you need. Some files, as you will see below, even have double extensions.

Collecting separate files into an archive does not compress the files. It only makes sense, then, that after you create an archive you might want to compress it. Thus, you will find many files that have extensions of **.tar.Z**, for example, **game.tar.Z**.

This means that a group of files was collected into an archive which was then compressed. When you download such a file, you must first use **uncompress** to restore the archive, and then use **tar** to restore the original files. If you forget what order to perform these operations, just look at the file name and work from right to left.

Later in the chapter, we will explain the details of using both **compress** and **tar**, as well as the other common compression and archiving programs that you may need to use.

Hint

On a DOS-based PC, the name of a file consists of a "filename" and an optional "extension". The filename can be no longer than eight characters, while the extension can be no longer than three characters. For this reason, compressed tarfiles that are meant to be used with DOS are often given an extension of **.taz**. For example, you might see a file named **game.taz**. This is because a name like **game.tar.Z** is not allowed with DOS. When you see such a file, the intention is that you will download it to your PC, and then uncompress and unpack it.

If you are at all involved with the world of PCs, there is another program you should know about. This program, called **pkzip**, both archives and compresses at the same time. To uncompress and unpack, you use a program called **pkunzip**. (The initials **pk** stand for Phil Katz, the program's developer.)

The **pkzip** program creates files with an extension of **.zip**, for example, **view.zip**. Such files are often referred to as *zipfiles*. We will discuss zipfiles and what to do with them later in the chapter.

File Types Used with Usenet and Mail

Broadly speaking, any file can be classified as text or binary. Text files contain ordinary characters: letters, numbers, punctuation, and so on. You can use these files directly as they are. For example, you can display a text file using a paging program (such as **more**, **pg**, or **less**), or edit a text file using a text editor (such as **vi** or Emacs).

Another name for text files is ASCII files. This is because the characters are stored according to a system called the ASCII code. (ASCII stands for the American Standard Code for Information Interchange.)

Binary files contain data that is meant to be used by a particular program. Here is a well-known example. Many people use Anonymous FTP and Usenet to share pictures that can be displayed on a computer screen. These pictures are stored as binary files and must be displayed by a picture-viewing program that understands the file format. If you were to look at a picture file with a paging program—which is designed to display text files—you would see gibberish.

Here is another example. People also share binary files that contain sounds that can be played back on computers with the appropriate hardware. In this

case, you need a sound program that understands the binary file format being used. More boring examples are word processors, spreadsheets, and databases that store data in binary files according to their own particular format.

In Chapter 5, we discussed text and binary files in more detail. At the time, we explained that mail messages can contain only text data, not binary data (unless you use a special protocol called MIME: Multipurpose Internet Mail Extensions). Similarly, we saw in Chapter 10 that Usenet articles are also composed of ordinary text.

This poses a problem: How can you mail a binary file to someone? And how can you post a binary file—such as a picture—to a Usenet newsgroup?

The solution is to convert the binary file to ASCII text. Then, you can mail it or post it as a Usenet article. Later, someone else can take the text file and convert it back to binary.

There are two programs that are commonly used to convert files from binary to text. The first is named **uuencode**. We refer to the text file that **uuencode** creates as a *uuencoded file*. To convert a uuencoded file back to binary, we use **uudecode**. (The names were chosen because these two programs were developed as part of UUCP, a system that can be used to copy files from one Unix computer to another. UUCP stands for "Unix to Unix copy".)

When uuencoded files are stored on an Anonymous FTP server, they are usually given an extension of **.uue**, for example, **file.uue**. Such files are sometimes compressed, in which case you will see a file name like **file.uue.Z**.

The other program that is commonly used to convert binary files to text is called **btoa** (binary to ASCII). We refer to the text file that **btoa** creates as a *btoa file*. There is no special extension used to identify such files.

We will discuss the **uuencode/uudecode** and **btoa** programs later in the chapter.

The last program that we will mention here is **shar**. It is used to create archives that are suitable for mailing or posting to Usenet. Files created by **shar** are called *shar files*. When shar files are offered for Anonymous FTP, they are usually given an extension of **.shar**, for example **file.shar**. Sometimes, such files are also compressed, so you will see file names like **file.shar.Z**.

The advantage of using a shar file is it is especially easy to unpack. The shar file itself forms an executable program that, when run, unpacks itself and re-creates the original files. The program is in the form of a Bourne shell script and you can run it by using the **sh** command. (**sh** is the name of the Bourne shell.) There is also a program named **unshar** that you can use for unpacking. We will explain how it all works later in the chapter.

For reference, Table 18-1 contains a summary of the file types we have mentioned so far. As an Internet user, these are the basic file types and programs that you should understand. If you do not use a PC, you can probably ignore zipfiles.

File Type	Example	Principal Programs
compressed file	`file.Z`	compress, uncompress
tarfile	`file.tar`	tar
compressed tarfile	`file.tar.Z`	tar, compress, uncompress
uuencoded file	`file.uue`	uuencode, uudecode
compressed uuencoded	`file.uue.Z`	uuencode, uudecode, compress, uncompress
btoa file	–	btoa
zipfile	`file.zip`	pkzip, pkunzip
shar file	`file.shar`	shar, sh, unshar
compressed shar file	`file.shar.Z`	shar, sh, unshar, compress, uncompress

Table 18-1. *Summary of File Types and Related Programs*

If you would like an exhaustively comprehensive list with every file type known to civilized man, you can download the file utility document we mentioned earlier. FTP to **ftp.cso.uiuc.edu** and look in the directory **/doc/pcnet** for a file named **compression**.

Hint

Before you download a binary file via FTP, don't forget to prepare by entering the **binary** command (see Chapter 16). If you do not use this command, the FTP program may assume that you are transferring an ASCII text file, and your file may be downloaded improperly.

Compressed files (**.Z**), tarfiles (**.tar**), and zipfiles (**.zip**) are always binary.

Uuencoded files (**.uue**), shar files (**.shar**), btoa files, and tarmail files (which we will meet later in the chapter) are always text.

Files That Hold Documents

There are two types of documents that are commonly found on Anonymous FTP hosts. First, there are regular text files. Such files often have an extension of **.txt**,

for example, **document.txt**. Once you download a text file, you can view it with any text editor or paging program.

Another common type of document file has an extension of **.ps**, for example, **document.ps**. These are Postscript files and are meant to be printed on a Postscript printer. Postscript files can contain pictures as well as characters.

Technically, Postscript files are text files, and it is possible to look at them with a text editor. However, the data is stored in Postscript format and you won't be able to make any sense out of it. If you need to look at a Postscript file, you either have to print it on a Postscript printer, or use a program that can display Postscript files on your screen.

Compressed Files (.Z)

Say that you have downloaded a file named **recipe.Z** via Anonymous FTP. Since the file name ends with a **.Z** extension, you know that it has been compressed with the **compress** program. To uncompress it, you use the **uncompress** program. Here is what to do.

Simply enter the name of the program followed by the name of the file:

```
uncompress recipe.Z
```

The **uncompress** program always expects file names to end in **.Z**, so you do not actually have to type the **.Z**. For example, the following two commands are equivalent:

```
uncompress recipe
uncompress recipe.Z
```

Either of these commands will uncompress the file named **recipe.Z**. A new uncompressed file, named **recipe**, will be created. If a file with this name already exists, it will be replaced, so be careful. The old file, **recipe.Z**, will be removed automatically.

If you would like **uncompress** to display a message showing you what it did, use the **-v** (verbose) option after the name of the command:

```
uncompress -v recipe
uncompress -v recipe.Z
```

In this case, you will see an informative message like:

```
recipe.Z:  -- replaced with recipe
```

If you would like to display the contents of a compressed file without actually uncompressing it, you can use the **zcat** program:

```
zcat recipe.Z
```

Again, you can leave off the **.Z**:

```
zcat recipe
```

The **zcat** command will show you what is in a compressed file without changing the file itself.

If the uncompressed file is long, most of it will scroll off your screen before you have a chance to read it. To look at the file, one screenful at a time, you can send the output of **zcat** to a paging program, such as **more**, **pg**, or **less**. To do so, follow the **zcat** command with a | (vertical bar) character, followed by the name of your favorite paging program. For example:

```
zcat recipe | more
```

Remember, though, you can only use **zcat** with a compressed text file. If you have a compressed binary file (such as **file.tar.Z**), you must uncompress and then restore the original files before you can view them.

When you use the | character in this way, we say that you *pipe* the output of one program to another. For more details, see a good Unix book. We recommend *A Student's Guide to Unix*, by Harley Hahn (McGraw-Hill).

Hint

The **zcat** program was named after the Unix **cat** command which, in its simplest form, displays the contents of a file with no added processing. The name **cat**, which comes from the archaic word "catenate", to join in a chain, was chosen because the **cat** program is also used to combine files.

To compress a file, you use the **compress** command. Type the command, followed by the name of the file. The **compress** program will create the new **.Z**

file for you and automatically remove the original file. For example, to compress a file named **new-recipe**, enter:

```
compress new-recipe
```

When the command finishes, you will have a new file named **new-recipe.Z**, while the original file **new-recipe**, will be removed.

If you would like **compress** to display a message showing you what it did, use the **-v** (verbose) option after the name of the command:

```
compress -v new-recipe
```

You will see an informative message like:

```
new-recipe: Compression: 53.95% -- replaced with new-recipe.Z
```

If you would like more information about using **uncompress**, **zcat**, or **compress**, check with a good Unix book. You can also display the official description of these commands by using the man command to display entries from the online Unix manual. (The online Unix manual is a built-in facility that can display information about any Unix command.) On most systems, these three programs are documented together, so you only need to use the single command:

```
man compress
```

Hint

On many Unix systems, you will find a pair of programs named **gzip** and **gunzip** which were written to be a replacement for the standard **compress** and **uncompress** programs. The **gunzip** program is powerful. It can restore both **.Z** files and **.zip** files. (We discuss **.zip** files later in the chapter.) The **gzip** program has its own type of compression that produces **.gz** files. Thus, if you download a file with an extension of **.gz**, you will need **gunzip** to unpack it.

If your system has **gzip** and **gunzip**, you should be able to display the documentation by using the command:

```
man gzip
```

You will see that there are actually three programs: **gzip**, **gunzip**, and a new version of **zcat**.

Tarfiles (.tar)

When you download a file whose name has an extension of **.tar**, you know that you have a tarfile. This is an archive that must be unpacked by using the **tar** command. When you unpack such a file, you will usually find multiple files, and sometimes directories. (We discuss Unix files and directories in Chapter 16.)

For this reason, it is a good idea to make a new directory to hold the tarfile before you unpack it. This ensures that all the unpacked files and directories will be segregated from your other files. In this section, we can't go into all the details of creating directories—you will need to consult a Unix book; however, we will show you an example.

Suppose you have downloaded a game via Anonymous FTP. The game consists of a number of program and documentation files that are contained in a single tarfile named **game.tar**. Here is what to do, once you have downloaded the file.

Start by creating a new directory for the game:

```
mkdir game
```

Next, move the tarfile to this directory:

```
mv game.tar game
```

Now, change your working directory to be the new directory:

```
cd game
```

Hint

In Unix, you can enter more than one command at a time by separating them with semicolons. Thus, you can perform all three of the previous commands in one line by entering:

```
mkdir game;  mv game.tar game;  cd game
```

At any time, you can display the name of your working directory by using the **pwd** (print working directory) command.

You are now ready to unpack the tarfile. Use the **tar** command, followed by the options **-xvf**, followed by the name of the file. For example:

```
tar -xvf game.tar
```

The **x** tells **tar** that you want to extract files. The **v** (verbose) tells **tar** to display extra information about each file as it is extracted. And the **f** indicates that the name of the tarfile follows directly.

After you unpack an archive, it is a good idea to list the new files and directories. To do so, use the **ls** (file list) command with the **-l** (long listing) option:

```
ls -l
```

The final consideration is that **tar** will not remove the original tarfile. You will have to do this yourself, by using the **rm** command:

```
rm game.tar
```

To summarize, the following series of commands will process an archive named **game.tar**. The commands create a new directory, move the tarfile to that directory, change to the directory, unpack the archive, list all the new files, and then remove the original tarfile:

```
mkdir game
mv game.tar game
cd game
tar -xvf game.tar
ls -l
rm game.tar
```

If you would like to see what is inside a tarfile without actually unpacking it, you can use the **t** (table of contents) option instead of **x** (extract):

```
tar -tvf game.tar
```

Hint

The **tar** command is unusual in that the options do not need to be preceded by a - (hyphen) character, as is the case with most Unix commands. Thus, the following two commands are equivalent:

```
tar -xvf game.tar
tar xvf game.tar
```

You will often see **tar** commands written this way, so don't be confused.

We believe that it is better to get in the habit of using the - character to precede options, because that is the convention that you must follow with virtually all other Unix commands.

To create an archive of your own, use the **tar** command with the **-cvf** options (**c** for create). Type **tar -cvf**, followed by the name you want for the archive, followed by the names of the files you want to include in the archive. For example, let's say that you want to collect three files named **groatcakes**, **chicken-soup**, and **brownies** into an archive named **cookbook.tar**. Use the command:

```
tar -cvf cookbook.tar groatcakes chicken-soup brownies
```

If you would like more information about using **tar**, check with a good Unix book or use the **man** command to display the **tar** entry from the online Unix manual:

```
man tar
```

Compressed Tarfiles (.tar.Z)

It is common to find tarfiles that have been compressed. Such files will have a double extension of **.tar.Z**. To use such files, all you need to do is follow the steps we have already outlined for uncompressing and unpacking. Just remember to uncompress first.

Say, for example, you download a copy of a book via Anonymous FTP. The various chapters of the book have been collected into a tarfile named **book.tar**.

This tarfile has been compressed into a file named **book.tar.Z**, which is the file you download. You need to do the following:

1. Create a new directory for the book.
2. Move the compressed tarfile to this directory.
3. Change your working directory to be this directory.
4. Uncompress the compressed tarfile.
5. Unpack the tarfile.
6. List the new files.
7. If it looks okay, remove the tarfile.

Here are some sample commands that do the job:

```
mkdir book
mv book.tar.Z book
cd book
uncompress book.tar.Z
tar -xvf book.tar
ls -l
rm book.tar
```

Hint

Remember, **uncompress** will remove the compressed file, but **tar -xvf** will not remove the tarfile.

Uuencoded Files (.uue)

A uuencoded file is a binary file that has been encoded as a text file by the **uuencode** program. You will most commonly see uuencoded files posted to Usenet newsgroups or sent as mail messages. This is because both Usenet and mail use text files, and a binary file must be encoded as text before it can be sent.

You will also see such files offered on Anonymous FTP hosts. There is no standard file extension (like **.Z** or **.tar**) that is always used to identify uuencoded files, but you will often see the extension **.uue**.

To decode a uuencoded file, you use the **uudecode** command. All you need to do is type the command followed by the name of the uuencoded file. For example:

```
uudecode picture
```

The **uudecode** program will read the uuencoded file and re-create the original binary file. When a binary file is uuencoded, its name is placed right into the uuencoded file. When the **uudecode** program reads this file, it looks for this name. Thus, **uudecode** always knows what name to give to the newly restored binary file.

Here is an example. You download a file called **harley.uue** via Anonymous FTP. According to some documentation in another file, **harley.uue** contains a binary file named **rick**. To restore the binary file, you enter:

```
uudecode harley.uue
```

Once **uudecode** is finished, you will have a binary file named **rick**. The **uudecode** program does not remove the original uuencoded file, so you must do it yourself. In this case, you would use the command:

```
rm harley.uue
```

When you read Usenet newsgroups that contain pictures, you will find that most of the uuencoded pictures are so large that they are broken into parts. Each part is posted as a separate Usenet article. Later in the chapter, we will show you how to put these parts together and restore the original binary file.

Btoa Files

Like **uuencode**, **btoa** (binary to ASCII) is a program that encodes a binary file as a text file. Btoa files are used mostly to send binary files through the mail. You use **btoa** to encode a file which you mail to a friend. At the other end, your friend saves the message to a file, and then uses **btoa** to restore the original file.

To encode a binary file to text, you use the **btoa** command followed by the name of the binary file, followed by the name you want to use for the text file.

For example, say that you have a binary file named **harley** that you want to mail to a friend. You can use the command:

```
btoa harley temp
```

You can now mail the file **temp** to your friend.

Hint

You can send the output of the **btoa** command directly to the **mail** command, which eliminates the need for a temporary file. For example, say that you want to mail the binary file **harley** to a friend whose address is **addie@nipper.com**. You can use:

```
btoa harley | mail -s 'btoa file of harley' addie@nipper.com
```

The | (vertical bar) character tells Unix to pipe the output of **btoa** directly to the **mail** program. The **-s** option tells **mail** what subject to put on the message. For more details on using **mail**, see Chapter 6.

To decode a btoa file, use the **btoa** command with the **-a** option, followed by the name of the encoded file. You do not need to specify a name for the binary file; it was encoded into the text file and **btoa** will find it automatically.

Thus, suppose you have received the mail message from the previous example. You save this message to a file named **temp**. Now, to recover the binary file, you can use:

```
btoa -a temp
```

When this command is finished, you will have the original file named **harley**.

Hint

If you receive a btoa file as a mail message, you will have to save it as a file. This new file will have header lines at the beginning (see Chapter 5). There is no need to remove these header lines before you decode the file. The **btoa** program will skip over them automatically.

The old version of **btoa** did not use **btoa -a** for decoding. Rather, it used a separate command named **atob** (ASCII to binary). If your system has the old **btoa**, you will have to use the **atob** command to restore the original file. For example:

```
atob temp
```

If this is the case, you might ask your system manager to install the new version of **btoa**.

Mailing Compressed Archives with Tarmail

As we mentioned, one of the most common uses of **btoa** is to encode a binary file as text so it can be mailed. In particular, **btoa** is often used to encode compressed tarfiles. For example, say that you have a number of chapters of a book in separate files that you would like to mail to a friend whose address is **ron@sigstar.com**. These chapters are named **ch1**, **ch2**, **ch3**, and **ch4**. (It's a small book.)

First, use the **tar** command to collect all the files into an archive named **book.tar**:

```
tar -cvf book.tar ch1 ch2 ch3 ch4
```

Next, compress the tarfile:

```
compress book.tar
```

Finally, encode the file from binary to text and mail it:

```
btoa book.tar.Z | mail -s 'Tarfile: Hot new book' ron@sigstar.com
```

At the other end, your recipient can uncompress and unpack the message to recover the original files. (Of course, he will need the **btoa** program.)

If you find this sort of thing handy, there is a program that will do it all for you in one step: collect a group of files into an archive, compress the archive, and

then convert it to a text file suitable for mailing. The name of the program is **tarmail**. To use it, you type the name of the program, the address of the recipient, a subject for the message, and the names of all the files you want to send. For example:

```
tarmail ron@sigstar.com 'tarmail book' ch1 ch2 ch3 ch4
```

When you receive a tarmail file through the mail, all you have to do is save it and then use **untarmail** to restore the original files. Simply type **untarmail**, followed by the name of the file.

For example, let's say that our friend receives our message and saves it to a file named **temp**. All he needs to do is enter:

```
untarmail temp
```

The program will restore the original files, in this case, **ch1**, **ch2**, **ch3**, and **ch4**. When **untarmail** is finished, it will delete the tarmail file for him.

Shar Files (.shar)

Shar files are used to collect text files into an archive. The archive is in the form of a shell script so it is easy to unpack.

Now, in order for any of this to make sense, you need to know what a shell script is, so let's take a moment to talk about shells.

In Unix, the program that reads and processes your commands is called the *shell*. When you enter, for example, the **date** command, the shell reads it and figures out what to do. In this case, the shell would execute the **date** program. When you enter the **tar** command, the shell executes the **tar** program, and so on.

Within the world of Unix, there are a number of different shells, the most popular being the C-Shell and the Korn shell. (You are probably using one of these shells, even if you don't know it.)

The original shell was called the Bourne shell. The Bourne shell is considered to be somewhat primitive these days and is not most people's first choice. Still, it is readily available on all Unix systems.

Each shell is itself a program that can be executed, just like any other program. The name of the C-Shell program is **csh**; the name of the Korn shell program is **ksh**; and the name of the Bourne shell program is **sh**. (The Bourne shell was the original shell, so it has the simplest name.)

Hint

You may be wondering, where do the names of the shells come from? The C-Shell was developed by Bill Joy, who named it after the C programming language because they share some features. The Korn shell was named after its developer, David Korn. The Bourne shell was named after its developer, Steven Bourne.

Now, one important aspect of the shell is that it can accept not only commands that you type in at your keyboard, but commands that are stored in a file. For example, you can tell the shell: "I have a file named **harley** that contains shell commands. Read that file, line by line, and execute those commands just as if I had entered them from the keyboard." Such a file is called a *shell script*.

Each shell offers a number of commands that can be used to create complex and powerful shell scripts. Thus, it is possible to write shell scripts that do many different things. For example, the **tarmail** and **untarmail** commands that we discussed earlier are actually shell scripts.

There are several ways to execute a shell script. The simplest way is to type the name of the shell that you want to run the script, followed by the name of the file that holds the script. For example, to have the Bourne shell (whose name is **sh**) run a script named **harley**, you can enter:

```
sh harley
```

Although you can run scripts with the C-Shell or Korn shell, most people use the Bourne shell, because it does a good job of executing scripts, and it is available everywhere. (However, as we mentioned earlier, people generally prefer to use the C-Shell and Korn shell as their primary command processor: the program that reads the commands they type at the keyboard.)

By now, it should make sense when we say that a shar file is actually a shell script. When you execute a shar file, it sends commands to the shell that, when carried out, re-create the original files that were archived. The beauty of this system is that you can archive a group of files into a shar file and mail it to anyone (or post it to Usenet). You are guaranteed that the recipient can unpack the files because everyone who uses a Unix system has access to the Bourne shell.

Thus, you can understand when we tell you that the name "shar" stands for "shell archive".

To create a shar file, you use the **shar** command. Type the name of the command, followed by the names of the files you want to archive, followed by a > (greater than) character, followed by the name of the file you want to hold the archive. For example, say that you want to create an archive named **recipes.shar**

to hold three files named **brownies, chicken-soup,** and **groatcakes**. You can then mail **recipes.shar** to a friend who appreciates good food.

Use the command:

```
shar brownies chicken-soup groatcakes > recipes.shar
```

The > character tells Unix to store the output of the command in the specified file. When you do this, we say that you are "redirecting the standard output to a file". For more details, see a good Unix book.

An alternate way to specify the name of the output file is to use the **-o** option after the command name. Type **-o** followed by the name of the output file. Thus, the following two commands are equivalent:

```
shar brownies chicken-soup groatcakes > recipes.shar
shar -o recipes.shar brownies chicken-soup groatcakes
```

Either of these commands will produce a shar file named **recipes.shar**.

Hint

One important limitation of shar files is that they can contain only text files. If you want to create a shar archive to hold binary files, you will have to use **uuencode** or **btoa** to encode the binary files as text files. However, in such cases, it is a lot easier to use **tar**, which can handle binary files directly. You can then encode the tarfile as a text file before you mail it.

If you look inside a shar file, you will see a message similar to the following one at the beginning of the file:

```
# This is a shell archive.  Remove anything before this line, then unpack
# it by saving it into a file and typing "sh file".  To overwrite existing
# files, type "sh file -c".  You can also feed this as standard input via
# unshar, or by typing "sh <file", e.g..  If this archive is complete, you
# will see the following message at the end:
#               "End of shell archive."
# Contents:  brownies chicken-soup groatcakes
# Wrapped by harley@fuzzball on Mon Jun 28 11:21:08 1993
```

Here is how to unpack a shar file that you see on Usenet or that you receive through the mail. First, save it to a file. Next, use a text editor and remove all the lines before the top line of the actual archive (as described above). If you have saved a Usenet article or a mail message, there will be header lines that you will have to remove. Finally, enter a command that consists of **sh** followed by the name of the file. The Bourne shell will read the shar file, execute its commands, and unpack the archive.

Here is an example. You are reading a Usenet newsgroup and you come across an article that contains a shar file. This shar file contains a number of recipe files, so you save the article to a file named **recipes.shar**.

Now, using a text editor (such as **vi** or Emacs), you remove all the header lines at the beginning of the file. To unpack the shar file, you enter the command:

```
sh recipes.shar
```

The original files will be unpacked for you.

You may have noticed one problem with this system. What if you don't know how to use a text editor? The solution is that you can use a program named **unshar** to unpack a shar file. The **unshar** program will look for and ignore header lines at the beginning of a file, so you don't have to remove them yourself. Thus, in the last example, we could have used:

```
unshar recipes.shar
```

Some versions of the **unshar** command save the header lines to a file with an extension of **.hdr**. In our example, this file would be called **recipes.shar.hdr**. If you want to tell **unshar** to not save the header lines in this way, you can use the **-n** option:

```
unshar -n recipes.shar
```

Not all Unix systems have the **shar** and **unshar** commands. However, the beauty of the scheme is that, even if you do not have these commands, you can still unpack a shar file, because all Unix systems do have the Bourne shell. Of course, if you do not have **unshar**, you will need to use a text editor. (This is why, in Chapter 1, we recommended that you learn some Unix.)

When you unpack a shar file, the original shar file is not removed. You will have to do this explicitly. For example:

```
rm recipes.shar
```

You may also have to remove the file that contains the header lines:

```
rm recipes.shar.hdr
```

Zipfiles (.zip)

If you use a PC, you should know that the most common programs for compressing and archiving PC files are **pkzip** and its sister program **pkunzip**. The **pkzip** program creates a compressed archive called a *zipfile*. The **pkunzip** program reads a zipfile and restores the original files. Thus, **pkzip** is both an archiver and a compression program (sort of like a combination of **tar** and **compress**). When we use **pkzip** to archive and compress a collection of files, we say that we are *zipping* the files. When we use **pkunzip** to unpack and uncompress, we are *unzipping*.

The PC version of the **pkzip** program is *shareware*. This means that the program can be freely distributed and used by anyone for free. However, if you like the program, you are urged to send a small fee to the company that develops **pkzip** (their name is Pkware). You will then become a registered user and receive certain advantages, such as a printed manual.

Aside from **pkzip** and **pkunzip**, there are programs that run on non-PC computer systems that can create and process zipfiles. Still, the most common zipfile program is the original **pkzip** for PCs. For this reason, you will find that many of the PC files that are available by Anonymous FTP are stored as zipfiles. Thus, as a PC user, it will be a common occurrence for you to download an Anonymous FTP file that needs to be unzipped.

In order to do so, you must follow two steps. First, you must get the file to your PC. (We will discuss how to do this in the next section.) Second, you must use the **pkunzip** program to unzip the file.

This is easy. On your PC, type the command **pkunzip** followed by the name of the zipfile. All zipfiles have an extension of **.zip**, but you can leave this out when you enter the **pkunzip** command. For example, the following two PC commands will both unpack and uncompress a zipfile named **recipes.zip**:

```
pkunzip recipes.zip
pkunzip recipes
```

Although zipfiles can be used to hold any type of data, they are commonly used to hold software packages. For example, later in the chapter, we will talk about software that you can use to view pictures on your PC. Such software

involves a whole package of programs and documentation files. To make such a package easy to download, the files are zipped into one tidy zipfile, which is then distributed via Anonymous FTP. As a user, all you need to do is download the zipfile and unzip it.

Hint

Most zipfiles contain a collection of files. Thus, before you unzip, it is a good idea to create a new directory to hold the zipfile. That way, all the unzipped files will be isolated in their own directory.

Among other advantages, this makes it easy to check the new files for viruses before you use them (something you should always do before you use downloaded PC software).

You say that you don't have an anti-virus program? Download one for free using Anonymous FTP. To find such programs, use Archie and search for **virus**.

The **pkzip/pkunzip** programs come with a lot of documentation, and we won't go into all the details here. What we will do is tell you how to get these programs for yourself. There are two ways. First, if your computer has a modem, you can dial the Pkware BBS (bulletin board system) at (414) 354-8670 and download the software (free) directly from the company.

Second, you can use Anonymous FTP and download the software from the Internet. To find an Anonymous FTP host, use Archie with a **regex** search and look for **^zip**. The ^ (circumflex) character tells Archie to match only those items that start with the letters **zip**. (We explain how to use Archie in Chapter 17.)

We mentioned earlier that many PC software packages are distributed as zipfiles. You might ask, is that how **pkzip** itself is distributed? The answer is no. After all, until you have **pkzip** up and running, it is impossible to unzip anything.

For this reason, **pkzip** is distributed as a *self-extracting archive*. This is an executable file that contains a collection of zipped files. When you run the program, it automatically unzips itself to restore the files. For example, at the time we wrote this chapter, the most current version of **pkzip** was distributed as a self-extracting archive named **pkz204g.exe**. When you run this program, it will restore the files that make up the **pkzip** product.

(You might wonder, if the Pkzip software is packaged in a file with a name like **pkz204g.exe**, how can you find it by using Archie to search for **^zip**? The answer is, the Archie search will find you a list of Anonymous FTP hosts that have a directory named **zip**. When you FTP to one of these hosts and look inside the **zip** directory, you will find the **pkzip** program.)

Hint

Program names that have numbers in them usually refer to the version of the program. For example, **pkz204g.exe** contains version 2.04g of **pkzip**. When you have a choice, always download the program that has the highest numbers.

If you see a program name that has the word **beta**, it means that this is a pre-release version that you can use at your own risk.

Hint

We mentioned earlier that there are programs for various different computers that can handle zipfiles. One of the best all round programs for Unix computers is **gzip**. It can not only handle zipfiles, but compressed (**.Z**) files as well.

Uploading and Downloading Files to Your PC or Macintosh

The second most frequently asked question asked by PC and Macintosh users is, how do I get Internet files to my computer? (The most frequently asked question is, how do I look at picture files?)

If you have the right software, downloading and uploading files between your personal computer and the Internet is easy. If you don't have the right software—or if it doesn't work properly—copying files will be impossible.

Hint

Quick review: Downloading refers to copying files from the remote host to your computer. Uploading refers to copying files from your computer to the remote host. To remember, think of the other computer as floating above you in the sky.

Before we start, let us recall that there are two ways to use a PC (or Macintosh) to access the Internet. In the simplest case, your PC might be

connected directly to the Internet. However, if you are working at home, you are probably using your PC to run a communications program. This program uses your modem to dial the phone number and connect to a remote Internet host. The program then emulates a terminal. You log in to the host with your userid (login name) and a password. All of this is explained in Chapter 3.

If your PC is connected directly to the Internet, files that you download via Anonymous FTP will come right to the PC. However, if you are using a communications program to connect to a remote host, the FTP files will be sent to that host. You will then have to copy them to your PC. That is what we will discuss in this section.

To copy files from one computer to another, we use what is called a *file transfer protocol.* In the PC and Macintosh world, the most common protocol is called Zmodem. Thus, in order to download and upload files, you need Zmodem software on both your computer and the remote host. All PC and Macintosh communication programs come with Zmodem functionality. What you need to ensure is that such a program exists on the host.

On Unix systems, the most common Zmodem program is named **sz** (the name means "send using Zmodem"). If your system has **sz**, there will be a companion program named **rz** ("receive using Zmodem"). You should be able to display the documentation for both these programs by using the command:

```
man sz
```

If your Unix system does not have a Zmodem program, you may find another file transfer protocol called Kermit. The program will probably be named **kermit**. We will not talk about Kermit here except to say that it is very slow and that you should use it only if you can't use Zmodem.

To download a file from the remote host to your PC or Macintosh, type **sz**, followed by either **-a** or **-b** (we will explain in a moment), followed by the name of the file. For example:

```
sz -a document
sz -b picture.gif
```

If you are downloading a text (ASCII) file, use **-a**. If you are downloading a binary file, use **-b**. If you are not sure what type of file you have, use **-b**.

Extra Important Hint

Never download a binary file without using **-b**. The **sz** program may look like it is working fine, but when the file gets to your PC, the data inside the file will be damaged. This is the most common answer to the question: Why can't I unzip the zipfile that I downloaded to my PC?

As soon as you enter the **sz** command, one of two things will happen. Your communication program may detect that a Zmodem transfer has been initiated and start transferring the file automatically. (Some communication programs can be configured to do this.) Or you will see a message similar to:

```
Sending in Batch Mode
```

At this point, you will have to tell your communication program to start a Zmodem download. How you do this varies from program to program, so you will have to read the documentation that came with your program. As an example, if you use Telix (a popular PC shareware communications program), you would press the PAGE DOWN key and then select "Zmodem" from a menu.

To upload a file from your PC or Macintosh to the remote host, start by entering one of the following commands (on the host):

```
rz -a
rz -b
```

(Remember to use **-b** if you are uploading a binary file.)

After you enter this command, you will see a message similar to:

```
ready. To begin transfer, type "sz file ..." to your modem program
```

You must now tell your communications program to begin a Zmodem upload. Again, how you do this depends on your program. With Telix, you would press the PAGE UP key, select "Zmodem" from a menu, and then enter the name of the file you want to upload.

As we said before, when it all works, it's simple. When it doesn't work, it will make you tear your hair out and wonder if Man is really Nature's last word.

Downloading and Viewing Picture Files

We don't know if you have ever read a Perry Mason book. If you have, you will know that every now and then, while Perry Mason is in court defending a client, the district attorney will ask a question and Perry Mason will jump up and say, "I object, the question is incompetent, irrelevant, and immaterial. The district attorney has not laid a proper foundation."

Lest we be accused of just such a transgression, we will now lay a proper foundation for the viewing of picture files.

Pictures are stored as binary files. There are various formats for storing such files, the two most common being GIF and JPEG. To display a picture, you use a program called a *viewer* that knows how to read the picture file and turn it into an image. To view a GIF file, you need a GIF viewer; to view a JPEG file, you need a JPEG viewer. You can obtain a viewer for free via Anonymous FTP.

GIF pictures are stored in files that have an extension of **.gif**. The name GIF stands for Graphics Interchange Format. JPEG pictures are stored in files that have an extension of **.jpg**. The name JPEG refers to the people that devised the standard: the Joint Photographic Experts Group.

GIF and JPEG files hold regular pictures. There is another common type of file, a GL file, that contains animated (moving) pictures. Such files have an extension of **.gl**. To view such pictures, you need a GL viewer.

Of all the Usenet newsgroups, the ones that use the most network resources are those devoted to swapping pictures. Near the end of Chapter 9, we showed a list of the most popular Usenet newsgroups. If you look at this list (Table 9-4), you see that the sixth most popular newsgroup is **alt.binaries.pictures.erotica** (right in between **alt.sex.stories** and **rec.arts.erotica**). As you can see, sex is one of the most popular subjects on Usenet, and, as you might guess, most of the pictures that are posted are erotic. However, there are newsgroups that contain other types of pictures, for example, **alt.binaries.pictures.misc**.

To view a picture, you must go through the following steps:

1. Use your newsreader program (see Chapter 11) to read the newsgroup that contains the type of pictures you like. For a list of newsgroups, see Appendix G. Look at the groups whose names start with **alt.binaries.pictures**.
2. Pictures are stored in uuencoded format. Most pictures are so large that they will be posted in several parts. Find a picture that appeals to you and save each part to a separate file.
3. Using a text editor, combine the files (in the correct order) into a single uuencoded file. As you do, you must strip out all the superfluous header lines. If you find yourself working with a lot of picture files, you can get a

utility program that will do all the combining and stripping automatically. For more details, read the frequently asked question list mentioned below.

4. Now, use **uudecode** to decode the uuencoded file. You should end up with a single **.gif**, **.jpg**, or **.gl** file
5. If necessary, download this file to the computer on which you want to view the picture. For example, you may want to download the file to your PC or Macintosh.
6. Use your viewer to look at the picture.

Hint

Remember that pictures are binary files. If you are downloading with **sz**, be sure to use the **-b** option.

There is a frequently asked question (FAQ) list that explains all about viewing pictures. (We described FAQ lists in Chapter 9.) The best way to introduce yourself to picture viewing is to read the FAQ list. It comes in three parts and is posted regularly to all of the picture newsgroups as well as **news.answers**.

If you would like to download the FAQ list by Anonymous FTP, use Archie to search for **pictures-faq**. This will be a directory that contains the three parts of the FAQ list. One place from which you can FTP these files is **rtfm.mit.edu**. Change to the directory named **/pub/usenet/news.answers/pictures-faq**, and download the files named **part1**, **part2**, and **part3**.

The last point that we will cover is where to find a picture viewer. There are several places to look. The best way to start is by reading **part3** of FAQ list. This article contains a large list of all different kinds of picture-oriented software and where to get them via Anonymous FTP. Look for the section that pertains to your computer and then look for the subsection called "Picture Viewers".

Along with the description of each viewer, you will also see the location of an Anonymous FTP host that carries the file. In addition, once you know the name of the file, you can use Archie to search for alternate sites.

Another place to look is the Usenet newsgroup **alt.binaries.pictures.utilities**. You will often see PC picture viewing programs posted here. Typically, the programs will be split into several postings. When you put together the postings and decode them, you will have a zipfile, which you can download to your PC.

One good place to know about is the Anonymous FTP host **bongo.cc.utexas.edu**. FTP to this host and change to the **/gifstuff** directory. You will see a number of sub-directories that contain files relating to various types of

computers. For example, if you are a PC user, you would change to the **msdos** sub-directory. Within **/gifstuff**, there is also a file named **ftpsites**. This file contains a list of a large number of Anonymous FTP hosts that carry picture-related files and programs.

Hint

Erotic pictures are available only in the Usenet newsgroups. There is no Anonymous FTP host that offers erotic pictures, so don't waste your time looking for one.

CHAPTER 19

Talking

In this chapter, we will show you how to have a computer-mediated conversation with another Internet user. You will be able to connect your computer to another person's computer and type messages back and forth. He will see what you type, and you will see what he types.

A handy and convenient communication tool... when it works. For those times when it doesn't work, we will show what might be wrong and give you hints on what to do.

Finally, we will tell you about a program that allows several people to have a group conversation, sort of like an Internet conference call.

393

Introducing the talk Program

The **talk** program allows you to connect your computer to someone else's computer, and then type messages back and forth. (Of course, both computers must be on the Internet.)

When you communicate in this way, we say that you are *talking* to that person. Of course, you are not talking in the usual sense—by speaking and listening—but you do see what the other person types as soon as he or she presses the key.

For this reason, the convention is to refer to such connections in an informal sense. For example, you might say that you had a "conversation" with someone yesterday. Or you might send a mail message that says, "If you have any problems, just talk to me."

The great thing about the **talk** program is that you can talk with people all over the world for free. For example, as we worked on this book in California, we often had conversations with a well-known computer expert in Austria.

As you talk to someone, the actual communication is handled by a program called a *talk daemon*. As we explained in Chapter 5, a daemon is a program that executes in the background, in order to provide a service of general interest. In this case, the daemon provides a talk service. In Chapter 8, we saw that a different daemon provides the Finger service.

To start a conversation, type **talk**, followed by the address of the person to whom you would like to talk. For example, say that your userid is **harley** and you are logged in to the computer named **fuzzball.ucsb.edu**. You want to talk to a friend whose userid is **rick**, who is logged in to the computer named **tsi.com**. Enter:

```
talk rick@tsi.com
```

The program will send a message to your friend's computer and tell him that you want to talk with him. Out of nowhere, he will see the following message on his screen and hear a beep.

```
Message from Talk_Daemon@tsi at 13:56 ...
talk: connection requested by harley@fuzzball.ucsb.edu.
talk: respond with:  talk harley@fuzzball.ucsb.edu
```

At this point, all he has to do is follow the instructions. In this case, he would enter the command:

```
talk harley@fuzzball.ucsb.edu
```

Once he does, the connection is complete and you can start talking.

If the person you are trying to reach is logged in, but is not responding, the **talk** daemon will keep sending him a message, about every ten seconds. As it does, you will see:

```
[Ringing your party again]
```

If you decide to give up, just press CTRL-C to stop the **talk** program.

Of course, you yourself may receive a **talk** request unexpectedly. If you do, just remember to enter the **talk** command exactly as you are requested in order to make the connection.

When you have finished your conversation, either person can quit by pressing CTRL-C. (If you are not using a Unix system, just press whatever key is used with your computer to stop a program.)

If you want to talk to someone who is logged in to the same computer as you, you need specify only a userid and not the full address. For example, say that you want to talk to someone on your computer whose userid is **mike**. You can use the command:

```
talk mike
```

The talk Screen

Once your **talk** daemon has made the connection with another person's **talk** daemon, you will see the message:

```
[Connection established]
```

and hear a beep.

The **talk** program will draw a horizontal line in the middle of your screen, dividing it into an upper half and a lower half. Whatever you type will be displayed in the upper half, whatever the other person types will be displayed in the lower half. Figure 19-1 shows a typical **talk** screen.

Hint

If you are using Unix and you notice that the output is displayed in a garbled fashion, it may be that you have not initialized your terminal type: that is, you have not told Unix what type of terminal you are using. Typically, you will see a line of characters going down the left-hand side of the screen.

We discuss terminal initialization in Chapter 3. If you are having this problem, check with a good Unix book or ask someone for help.

You can both type at the same time, and whatever you type will be displayed immediately on both screens. As you type, you can press BACKSPACE to correct the previous character. You can also use CTRL-W to erase an entire word and CTRL-U to erase an entire line. At any time, you can tell **talk** to re-display the entire screen by pressing CTRL-L. This can come in handy if an unexpected message—like someone else trying to talk with you—has appeared on your screen.

(You may have to experiment a little. With some computers, you have to use DELETE instead of BACKSPACE to erase a single character. Similarly, you may have to use CTRL-X instead of CTRL-U to erase an entire line.)

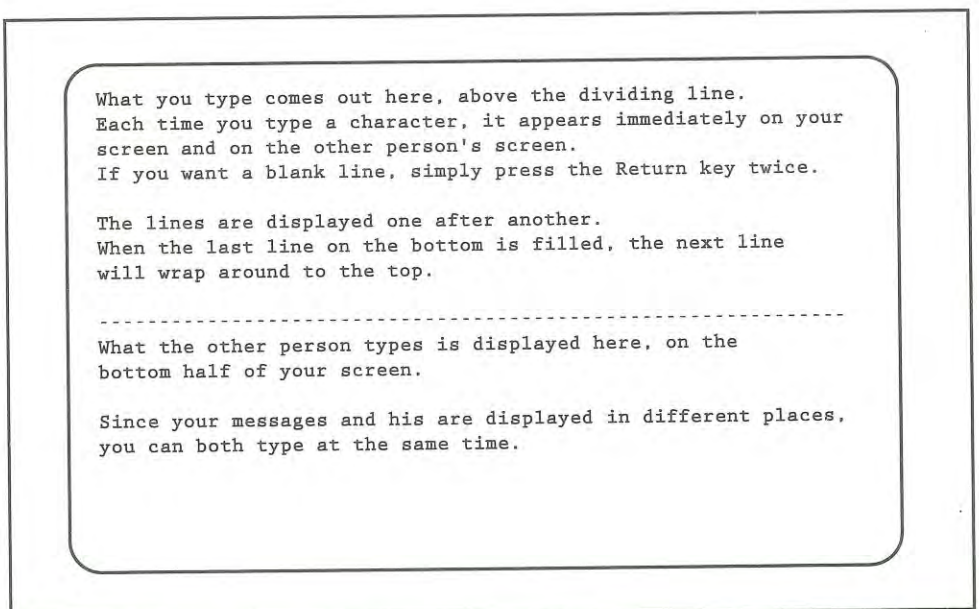


Figure 19-1. *A typical* **talk** *screen*

Hint

Because the other person can see each character as you type it, he will also see you backspace when you make corrections. Thus, do not type something offensive thinking that you can change it. The other person will see exactly what you type, as you type it.

Hint

One of the biggest mistakes that people make is pressing BACKSPACE repeatedly to correct a few words or the entire line. It is much faster to use CTRL-W and CTRL-U (or CTRL-X).

Talk as long as you like. When you are finished, either person can sever the connection by pressing CTRL-C. It is considered good manners to say goodbye and to wait for an acknowledgment before ending a conversation. When the **talk** program is finished, you will see:

```
[Connection closing. Exiting]
```

Guidelines for Talking

Having a conversation via **talk** combines the immediacy of a face to face encounter with the verbosity of written language. For this reason, you will notice two significant differences between talking by computer and talking by mouth.

First, talking by computer is a lot slower, because almost everybody can speak faster than they can type (not that this is always an advantage). Second, because you are not looking at the other person, the customary nuances and inflections that comprise so much of the non-verbal aspects of communication are missing.

We make up for these limitations in two ways. First, there are a number of abbreviations that are commonly used to speed things up. Some of these are shown in Table 19-1. These abbreviations are the spiritual descendants of the Morse code conventions used by ham radio operators since the 1920s.

Abbreviation	Meaning
BCNU	be seeing you
BRB	be right back
BTW	by the way
BYE	goodbye (I am ready to stop talking)
BYE?	goodbye? (are you ready to stop talking?)
CU	see you
CUL	see you later
FYI	for your information
FWIW	for what it's worth
GA	go ahead and type (I will wait)
IMHO	in my humble opinion
IMO	in my opinion
JAM	just a minute
O	over (your turn to speak)
OO	over and out (goodbye)
OBTW	oh, by the way
ROTFL	rolling on the floor laughing
R U THERE?	are you there?
SEC...	wait a second
WRT	with respect to

Table 19-1. *Common Communication Abbreviations*

The second important convention recognizes that it is easy to offend someone (or at least have your words misconstrued) when you are not talking in person. Over the computer, we are unable to use subtle body language to indicate that what we just said was debonair and ironic, and not really as dumb and insensitive as it seems.

Thus, when there is any doubt as to whether or not the other person might be offended, it is a good idea to use a smiley to show that you are just kidding. As we explained in Chapter 10, a smiley is a little face that we draw with punctuation characters. Here is the basic smiley:

```
:-)
```

(To see the face, tilt your head to the left.) You can use a smiley whenever you say something that might bother the other person, for example:

```
I'd feel the same way as you, if I had a
head that looked like a ripe eggplant :-)
```

If you want to see a few more smileys, look at Figure 10-3 in Chapter 10.



FUN TIP: For possibly the first time in your life, spelling does not count.

Because typing is so much slower than talking, it is perfectly acceptable to type as fast as you can and not worry about spelling mistakes. Nothing is worse than watching someone laboriously backspacing and correcting each word. Just go ahead and type and don't worry. Almost everthigg is undrstanable with kust a few worng lettrs and. no doubt,m the other pweosn will get the g9ist of thwta you are syaing.

Problems with Addresses and What to Do

There are several problems that might keep you from making a **talk** connection.

First, the person you want to talk to may not be logged in. If this is the case, you may see a message like:

```
[Your party is not logged on]
```

The solution, of course, is to try again later. You may also want to mail the person a message, asking him to talk to you when he logs in.

Second, it may be that the computer you are trying to reach is not on the Internet. Remember, there are some computers that can exchange mail with the Internet, but are not on the Internet itself. If you specify the address of a computer that is not on the Internet, you will see a message similar to:

```
elmer.com is an unknown host
```

You may also see either of these messages if you make a typing mistake when you enter the **talk** command. For example, say that you want to talk to a friend whose userid is **rick** on the computer named **tsi.com**. However, by mistake, you type:

```
talk ric@tsi.com
```

You will see the message:

```
[Your party is not logged on]
```

Do not misinterpret this message. It does not imply that the userid you typed is valid, but the person is not logged in. All it means is that no one with that userid is currently logged in. Thus, when you get this message, be sure to check your spelling and, if necessary, retype the command.

Similarly, if you enter:

```
talk rick@tsii.com
```

you will see:

```
tsii.com is an unknown host
```

At least this time, the message is more informative and your spelling mistake is obvious.

The next reason that your connection might not work is that you may be using a mail address that is an alias and not the person's actual address.

As we explained in Chapter 4, some organizations allow their users to use a simplified mailing address. For example, say that you have a friend named Melissa who has userid **wx1523** on the computer named **sdcc99.ucsd.edu**. For convenience, her organization allows her to receive mail at the address **melissa@ ucsd.edu**. When a message arrives, the local mail program looks up the name in a master list and sends the mail to the correct local address.

However, if you want to talk to your friend, you cannot use **melissa@ucsd .edu**, as this is only a mail alias. You must use the exact address of **wx1523@ sdcc99.ucsd.edu**. If you are not sure of the correct address, you can always mail her a message and ask.

The "Checking for Invitation" Problem and What to Do

So far, we have discussed problems that stem from an incorrect address, or from a person not being logged in. However, there is a problem that commonly arises even when you have the correct address and the person is logged in. The reason is that not all Internet computers can use **talk** to connect to one another.

There may be a time when you enter a **talk** command and see the following message:

```
[Checking for invitation on caller's machine]
```

This looks like a nice, benign message. It seems to imply that the other person has been notified that you want to talk, and you are just waiting for him or her to enter the appropriate **talk** command.

In reality, this message is highly misleading. What it usually means is that your **talk** program and the other **talk** program are incompatible. (More precisely, your daemons are incompatible.) The reason is a little technical, but here it is.

As you may know, computer data is stored in bytes, each byte holding a single character. Inside the computer, the bytes are collected into words. There may be two bytes per word, four bytes per word, or whatever. The question is, within each word, what order are the bytes stored in?

For example, say that you have a word that contains two bytes, call them A and B. Some computers store the bytes as AB, while other computers store the bytes as BA. Normally, the hardware takes care of all the details, so you don't have to worry about it. The trouble comes when one **talk** program tries to communicate with another one.

If you are using the original version of **talk**, communication can only be established between two computers that store bytes in the same order. Unfortunately, many Internet computers still use this version of **talk**, and when two incompatible computers try to connect, you get stuck with the "checking for invitation" message. In this situation, the only thing you can do is press CTRL-C and abort the command. (But see below for some more ideas.)

Technical Hint

The two schemes that we described are called *big-endian* ("big end first") and *little-endian* ("little end first"). Sun computers and Macintoshes are big-endian. VAX computers and PCs are little-endian.

This incompatibility is sometimes referred to as the *NUXI problem*, because the word "UNIX" on one computer would be stored as "NUXI" on another.

What this all means is that if you are using the original version of **talk**, you can only talk to people who are using compatible computers. To solve this problem, another version of **talk**, called **ntalk** (new **talk**), was developed. The **ntalk** program can talk to any other **ntalk** program, regardless of what computer it is running on. (In case you are interested, **ntalk** communicates via a facility called "sockets", which is not sensitive to byte order.)

Many system managers have installed **ntalk** on their machines, so if you are having trouble with **talk**, try entering the same command again using **ntalk**. For example:

```
ntalk rick@tsi.com
```

Still, there is a problem. Some system managers install **ntalk** under that name. When you want the new version, you use **ntalk**, when you want the old version, you use **talk**.

Other system managers install **ntalk** under the name **talk** (to make it more convenient!) and rename the original version to **otalk** (old **talk**). On those systems, you use **talk** to get the new version and **otalk** to get the old version.

Don't be confused. If you have problems using **talk** to connect to a particular computer, just experiment with **ntalk**, **talk**, and **otalk** until you find the combination that works.

The real solution to all of this silliness is to use a much better program called **ytalk**, which we will discuss later in the chapter. Not only will **ytalk** work with any other computer—whether it is running **talk**, **ntalk**, or **ytalk**—it allows more than one person to talk at the same time. The **ytalk** program is readily available via Anonymous FTP and, if you are having **talk** problems, you might prevail upon your system manager to install **ytalk**.



FUN TIP: What can you do if you and your friend do not have two compatible versions of **talk**? *You will have to find another way to get together.*

For example, you could agree to meet on Internet Relay Chat (IRC), which we discuss in Chapter 20. You could also use one of the Internet-accessible bulletin board systems (BBS), which we mention in the catalog. And, of course, you can always send messages by mail.

Refusing to Talk

There may be times when it is inconvenient to receive invitations to talk. For example, you may be editing an important file and you do not want your screen cluttered up with messages from the **talk** daemon. Or you may be a private person who simply does not want to talk to anyone else.

By using the **mesg** (message) command, you can tell your system to refuse or accept **talk** requests on your behalf. To refuse all such invitations, type **mesg** followed by **n** (no):

```
mesg n
```

To ask your **talk** daemon to show you such invitations, use **mesg** followed by **y** (yes):

```
mesg y
```

Normally, **mesg** is set to **y**, so you do not have to worry about it unless you want to cut yourself off from the outside world. To check your status, just enter the command by itself:

```
mesg
```

and **mesg** will display the current setting.

If you try to talk to a person who has set **mesg** to **n**, you will see a message like:

```
[Your party is refusing messages]
```

In such cases, about all you can do is mail a message.

One way to check if a person is accepting messages is to finger him (see Chapter 8). Some Finger servers will tell you whether or not a userid is accepting messages. Others will not; you will just have to try.

Here is an example. You have a friend whose address is **tln@nipper.com** and who has set **mesg** to **n**. You enter:

```
finger tln@nipper.com
```

As part of the output, you see the line:

```
Login name: tln  (messages off)  In real life: The Little Nipper
```

This shows you that this person is not accepting **talk** requests. You can also finger the **nipper.com** computer to show a summary of all the users who are currently logged in:

```
finger @nipper.com
```

When you do, one of the lines of output is:

```
tln      The Little Nipper    *p4  7 Tue 21:48
```

The details of reading such output are explained in Chapter 8. What we want you to notice is the * character. This tells you that this person is not accepting **talk** requests.

Hint

Before you try to talk to someone, you may want to finger their userid. Aside from checking if the person is accepting **talk** requests, you may also see if the person is logged in.

If the person is logged in, but you see that he has been idle for a long time (some Finger servers tell you this), it is likely that he will not respond to your **talk** request.

The ytalk Program

The **ytalk** program was written by Britt Yenne as a replacement for the standard Unix **talk** programs. The name **ytalk** stands variously for either "Yenne **talk**", or "Why talk?" depending on your state of mind.

The nice thing about **ytalk** is that it will work with both the old and new versions of **talk** that we discussed earlier in the chapter. If you are using **ytalk**, you never have to worry about what type of **talk** your friends are using.

Moreover, if you know other people who use **ytalk**, you can use it to hold a group conversation. At the end of this section, we will tell you how to get the **ytalk** program if it is not already installed on your computer.

The essential form of the **ytalk** command is the same as the **talk** command. Enter the command, followed by the address of the person you want to talk to:

```
ytalk rick@tsi.com
```

When you talk with one person, the screen divides into two parts, just like with **talk**. One important difference is that **ytalk** will show you the userid of the person to whom you are talking. Figure 19-2 shows such a screen. What you type is displayed on the top half. What the other person types is displayed in the bottom half.

If you want to talk with more than one person, all you have to do is specify more than one address as part of the command. For example, here is a command that sets up a five-way conversation:

```
ytalk  rick@tsi.com  mike  addie@nipper.com  tln@nipper.com
```

```
----------------------= YTalk Version 2.0 =--------------------
What you type is displayed here.

Just as with the regular talk command,
one can type at the same time as
the other person.



----------------------------= rick =---------------------------
What the other person types is displayed here.

He can type at the same time as you and
both parts of the screen will be updated
simultaneously.
```

Figure 19-2. *A* **ytalk** *screen with a two-way conversation*

In such cases, **ytalk** will divide the screen into as many partitions as necessary. In this example, the screen would look like Figure 19-3.

At any time, you can change the participants by calling up a special menu. To do so, press the ESC (Escape) key. You will see:

```
##################################################
# a) Add a new user to session                   #
# d) Delete a user from session                  #
# o) Output a user to a file                     #
# Your choice:                                   #
##################################################
```

You can now press either **a**, **d**, or **o**.

If you press **a**, you will be asked to enter the address of the person you want to add. Once you enter this address, **ytalk** will send them a message. When they respond, they will be incorporated into the conversation.

If you press **d**, you will be asked to enter the userid of the person you want to delete. In this case, you need only enter the userid and not the full address. Once you do, the person will be removed from the conversation.

If you press **o**, you will be asked to enter the name of a userid and then the name of a file. From then on, everything that is typed by that person will not only be displayed on your screen, but will be copied to the file as well. This option is useful when you want to save what someone is typing.

```
----------------------= YTalk Version 2.0 =--------------------
What you type is displayed here.

----------------------------= rick =----------------------------
What Rick types is displayed here.

----------------------------= mike =----------------------------
What Mike types is displayed here.

----------------------------= addie =---------------------------
What Addie types is displayed here.

----------------------------= tln =-----------------------------
What The Little Nipper types is displayed here.
```

Figure 19-3. *A* **ytalk** *screen with a five-way conversation*

In order to have a group conversation, all the people need to be using **ytalk**. However, if you have **ytalk**, you can still talk to people who are using the regular **talk** program. It's just that they will only be able to talk to you, and not to anyone else.

For example, say that you have **ytalk** and you want to talk with two friends who only have **talk**. If you connect to both of them, you will see a three-person conversation. Each of them, however, will only see the standard two-person **talk** screen. They will be able to talk with you, but not with each other.

What happens if you are in the middle of a conversation and a new person tries to talk to you? You will see a message like the following:

```
##################################################################
# Talk with mschuster@netsys.com?                                #
##################################################################
```

You can now press **y** (yes) to add the person to the conversation, or **n** (no) to turn them down. If you press **y**, **ytalk** will redraw your screen to add another partition.

When this happens, each person who is using **ytalk** will be asked if he wants to add the new person to his screen. He will see a message like:

```
##################################################################
# Import mschuster@netsys.com?                                   #
##################################################################
```

He can now answer **y** or **n** as he wishes.

As you can see, **ytalk** is an especially handy program. However, it is not a standard part of Unix, so there is a good chance that it may not be on your system. (Indeed, it will only be on your system if someone has deliberately installed it.)

If this is the case, you can download **ytalk** via Anonymous FTP. To find the program, you can use Archie. (We discuss Anonymous FTP in Chapter 16 and Archie in Chapter 17.)

When you use Archie, specify a **regex** search and look for the pattern **^ytalk**. For example, with an Archie client you would use the command:

```
archie -r ^ytalk
```

(The ^ character tells Archie to look for names that start with **ytalk**. If you do not do this, you will find some superfluous names that have **ytalk** in the middle.)

Once you download the program, you can install it if you know how to install programs. Otherwise, you will have to ask someone for help. (Show this page of the book to your system manager and ask nicely. He or she may agree to install **ytalk** as a public service.)

Hint

Whenever you look for a program on Anonymous FTP, make sure you get the newest version, because not all FTP hosts are up to date. You can often tell the version by looking for a number in the file name. For example, if you see a file named **ytalk-2.0.tar.Z**, you know that it holds version 2.0 of the program. (BTW, we explain how to deal with a **.tar.Z** file in Chapter 18.)



CHAPTER 20

Internet Relay Chat

In this chapter, we will introduce you to one of the most wonderful facilities on the entire Internet. You will learn how to join a permanent, 24-hours-a-day gathering, where people from all over the world carry on a multitude of conversations about everything and anything.

411

What Is Internet Relay Chat?

In Chapter 19, we discussed two ways for you to talk directly with other Internet users. You can use the **talk** program and talk to one person at a time, or you can use **ytalk** and talk to one or more people at the same time. What these two programs have in common is that your conversations are private, like a telephone call. You control who you talk to and only those people can take part in the conversation. A *chat facility* affords a different type of talking experience in which you talk to many people from all over the Internet.

Imagine a vast number of people mingling with one another, perhaps at a large party. You are standing with a group of people who are having several conversations. Whatever you say can be heard by everyone else in your group. You can also eavesdrop on other people's conversations. If you'd like, you can walk from one group to another, join in one of their conversations or just listen. You can also invite someone (or several people) over into a corner for a more private talk. And, if the need arises, you can whisper a private message in someone's ear.

This is the *Internet Relay Chat* or *IRC*.

IRC was developed in 1988 by Jarkko Oikarinen in Finland. The current version of IRC is called IRC II. Since its inception, IRC has become one of the most popular Internet resources and is frequented by people in many different countries. Here is how it works.

You use a client program to act as your interface. (We discuss the client/server relationship in Chapter 2.) Your client connects to an IRC server. You can then enter IRC commands, of which there are many. You can join a group of people or move from one group to another, talking and listening as you please.

Each IRC server is connected to nearby servers. Thus, all the IRC servers are connected (at least indirectly) with one another. So when you make contact, you are connected to a global web of IRC users, all of them talking. Twenty-four hours a day, you will find a vast number of people from many different countries, all connected to IRC, talking, talking, talking.

To bring order to what otherwise would be conversational chaos, IRC maintains a number of different *channels*. When you first connect to IRC, you choose which channel you want to join. You can now talk to all the other people on the same channel. When you feel like a switch, you can leave that channel and join another one. If you like, you can even join more than one channel at the same time.

At any particular time, you will find several thousand people, participating in hundreds of different channels. As you move from one channel to another, you will find people from all over the world. Most of the conversations will be in English, but many are in other languages.

To keep things from being too confused, IRC uses two conventions. First, every channel has its own name. Most channel names begin with a # character, for example, **#hottub**. Some channels are for discussing specific topics. Others are just for talking about whatever comes up. There are public channels, private channels, and secret "invisible" channels.

How does a channel get created? Very simply. Whenever you enter the command to join a channel, IRC checks to see if that channel already exists. If so, you will join it. Otherwise, IRC will create a new channel using whatever name you specified. At first, of course, you will be the only one on the channel, but you can wait for other people to join you. When the last person leaves a channel, IRC removes it.

The first person who joins (and thus, creates) a channel is given the status of *channel operator*. The operator has control over various aspects of the channel. For example, he or she can make the channel private, so that only people who are invited are allowed to join. An operator can also extend operator status for that channel to another person and set the topic for discussion.

Thus, IRC is more than people talking on public channels. You can set up your own channels whenever you want. For example, the President of the United States might send you a mail message and ask, "Would you like to join Hillary and me on channel **#algore** tonight at 8:30 PM, Eastern time?"



FUN TIP: If you would like to mail a message to the President, asking him to join you on IRC, his address is **president@whitehouse.gov**. *If he can't make it, Rush Limbaugh's address is* **70277.2502@compuserve.com**.

The other convention is that every IRC participant has a nickname. When you start a new IRC session, you specify a nickname, up to nine characters, which you can also change at any time. The only limitation is that you cannot choose a nickname that is currently in use. If you want a permanent nickname, you can register it with a special IRC database, so you can always be known by the same name.

As you might imagine, the world of IRC is filled with all manner of imaginative nicknames. However, nicknames offer more than whimsy: they allow people to participate freely and openly while still retaining their anonymity.

At all times, your IRC client will act as your window, showing you what everyone in your channel is saying. Your screen display will vary, depending on what client program you use. Figure 20-1 shows a typical screen.

The bottom line of the screen is where you type your input. In our example, we have typed:

```
Here is where you type, on the bottom line...
```

What you type is shown on this line, but is not actually transmitted until you press RETURN. This allows you to make corrections as you type.

Hint

IRC does not distinguish between upper- and lowercase letters when you type a command, a nickname, or a channel name.

```
<Dizney> So of course I told her to go ahead
<Davidg> So then what happened?
<Oinker> How can you say that, Sharkface?
<Dizney> How about you, Harley?
> I have never had that exact same experience
<Dizney> How do you know?
*** Nipper (tln@nipper.com) has joined channel #hottub
> Well, I'm not exactly sure
*** Addie (addie@fuzzball.ucsb.edu) has joined channel #hottub
<Nipper> Hi everyone
<Mindblow> Hello
<Addie> Hi Nipper, how are you?
<Davidg> Dizney, you never told us what happened
<Nipper> Addie, I am doing just fine, just ate dinner but
<Nipper> I accidentally knocked down a picture frame
<Nipper> and broke the glass.
<StreetWise> Where is everyone tonight?
*** Oinker has left channel #hottub
*** Guessme (whoami@silly.com) has joined channel #hottub
[1] 23:06 Harley (+i) on #hottub (+nt) * type /help for help
Here is where you type, on the bottom line...
```

Figure 20-1. *A typical conversation with Internet Relay Chat*

The second line from the bottom is a status line. In our example, it is:

```
[1] 23:06 Harley (+i) on #hottub (+nt) * type /help for help
```

In this case, we are using the nickname **Harley** and have joined the **#hottub** channel. We can also see the time (23:06, which is 11:06 PM), some technical information, and a reminder as to how to display help information.

The rest of the screen is given over to the actual conversations. As you watch the screen, you will see what is being typed by the people in your channel. Line after line will appear at the bottom of the display area. As new lines appear, the old ones scroll up. Thus, an IRC experience is one of watching many messages, one after the other.

You will sometimes notice that, if parts of the network slow down, people's replies are delayed and become unsynchronized, leading to various amusing and incoherent situations. In such cases, we refer to the delay as *lag*. Thus, you might see someone say, "There was so much lag, I thought you had forgotten all about me."

In our example, you can see several people talking at once. Each line is prefaced by the nickname of the person who typed the message. For example, the line:

```
<Nipper> I accidentally knocked down a picture frame
```

was typed by the person whose nickname is **Nipper**. The lines that you type do not have a nickname when they are displayed on your screen. Rather they have a single > character. For example, we typed the line:

```
> Well, I'm not exactly sure
```

Of course, on everyone else's screen, that same line will appear with your nickname. For example, if your nickname is **Harley**, everyone else would see:

```
<Harley> Well, I'm not exactly sure
```

Lines that begin with *** show informative system messages. For example:

```
*** Nipper (tln@nipper.com) has joined channel #hottub
```

You will often see people carrying on more than one discussion at the same time, and when things get busy it can be somewhat disorienting. Still, you will find that in a short time, you will synchronize with the IRC rhythm and it will all make sense. So much so, that unless you develop some self-discipline, you will find yourself spending more and more time chatting and less and less time participating in regular life (paying your bills, studying, washing dishes, buying computer books, and so on). Thus, before you get started, let us warn you:

> **Hint**
>
> IRC is addictive.

Learning About IRC

The best way to learn how to use IRC is to read something about it. Start with the IRC primer and the IRC tutorial, both of which are available by Anonymous FTP. You should also read the IRC frequently asked question (FAQ) list. If you want more technical information about IRC and its protocol, look at RFC #1459. (We discuss protocols in Chapter 2 and RFCs in Chapter 16.)

You can download the IRC primer by Anonymous FTP. Look in the following places for a file that begins with the name **IRCprimer**.

Host Name	Directory
`nic.funet.fi`	`/pub/unix/irc/docs`
`cs.bu.edu`	`/irc/support`
`coombs.anu.edu.au`	`/pub/irc/docs`

There are three versions of the file, as you will see when you look in one of these directories. The file with an extension of **.txt** is a regular text file. The one with an extension of **.ps** is a Postscript file. And the one with an extension of **.tex** is a Tex file. (Tex is a typesetting system.) If you download a file whose name ends in **.Z** you will have to uncompress it. (We discuss Anonymous FTP in Chapter 16 and file types in Chapter 18.)

You can download the IRC tutorial by Anonymous FTP. FTP to **cs.bu.edu** and change to the directory named **/irc/support**. Download the files that begin with

the name **tutorial (tutorial.1, tutorial.2**, and so on). You can use the command **mget tutorial.***.

The IRC frequently asked question list is posted regularly to several Usenet newsgroups, including **alt.irc** and **news.answers**. You can check these groups, or you can retrieve the document via Anonymous FTP. FTP to **rtfm.mit.edu** and look in the directory **/pub/usenet/news.answers** for a file named **irc-faq**. (We discuss FAQ lists in Chapter 9.)

More technical information about IRC can be found in RFC #1459. You can find it by using Archie to search for **rfc1459**. (We discuss RFCs in Chapter 16 and Archie in Chapter 17.)

Starting Your IRC Client

To use IRC, you start an IRC client program. The client will automatically connect you to an IRC server and the rest of the world.

It may be that you already have an IRC client (probably named **irc**) on your computer. If so, you should be able to connect to IRC just by entering **irc** followed by the nickname you want to use. For example:

```
irc Harley
```

If you have a Unix system, you can display the documentation for this command by using the **man** (online manual) command at the Unix prompt (not from within IRC):

```
man irc
```

If you do not have an IRC client, you have three choices. First, you can ask your system manager to install one. (Ask nicely. Some system managers feel that IRC is a waste of time.)

Second, you can download the client program from an Anonymous FTP host. The FAQ list will show you where to look. There are various clients available for different types of computer systems. If you download a client program, you will have to install it yourself. If you have any trouble, you may need to ask for help from your system administrator or a friendly computer nerd.

Hint

If you decide to download an IRC client, make sure you get the program for the client and not for the server.

Finally, there are public IRC clients that you can use via Telnet. Look in the IRC FAQ list for an address and then telnet to that address. You may have to specify a port number. (We discuss Telnet and port numbers in Chapter 7.) Once you connect, an IRC client program will start automatically. Conceptually, this is like telneting to an Archie server when you do not have an Archie client of your own.

Hint

The least preferred way to use IRC is by telneting to a public client. Such systems limit the number of users they will accept, and they are often busy. If you get at all serious about IRC, you will find it much more convenient to use a client program on your own computer.

Getting Started

Once you are connected to IRC, there are a great many commands that you can use. All IRC commands start with a / (slash) character. The / must be the first character on the line. As you type a command, it will appear on the bottom line of your screen. When you press RETURN, the command will be sent to IRC. Whenever you type a line that does not start with a / it is considered to be part of a conversation (a regular line) and will be sent to your channel.

The first command you should learn is the one to quit. It is:

```
/quit
```

To display a list of all IRC commands, enter:

```
/help
```

To display help information about a specific command, use **/help** followed by that command. For example:

```
/help join
```

As a new user, there are two special **/help** commands to use. To display information of interest to beginners, enter:

```
/help newuser
```

To display a short introduction to IRC, use:

```
/help intro
```

When you first connect to IRC, you will not be in a channel, and any conversation that you type will be discarded. The first thing you will want to do is decide which channel to join. To display a list of all the channels, enter:

```
/list
```

For each channel, you will see the name, the number of users, and the topic (if one is set).

Hint

Some commands—such as **/list**, **/who** *, or **/whois** * (both described later)—can display a long list of information that seems to go on and on. To stop the flow, you can enter the command:

```
/flush
```

However, have patience. This command does not always take effect right away. The best thing is to avoid using commands that produce so much output.

To make the channel list more manageable, you can ask to see only those channels that have a minimum number of participants, by using **-min** followed by a number. For example, to list only those channels that have at least fifteen people, use:

```
/list -min 15
```

You can also use **-max** to display channels that have no more than a specified number of participants:

```
/list -max 5
```

To display information about a particular channel, you can specify its name. For example:

```
/list #hottub
```

Aside from these variations, there are other options that you can use with **/list**. For more information, use **/help list**.

Once you decide what channel looks good, you can join by using **/join** followed by the name of the channel. Remember, most channel names start with a # character, for instance **#hottub**. Thus, to join this channel, you would enter:

```
/join #hottub
```

From now on, everything you type (except commands) will be sent to that channel. If you are composing a long message, it is not necessary to keep pressing RETURN. IRC will allow you to type up to 256 characters in a single message.

Hint

If you are not sure which channel to join, **#hottub** is a good place to start. Think of it as a large group of people, sitting around in a hot tub, talking about anything that enters their minds. (You will need to supply your own water.)

To leave a channel, type **/leave** followed by the name of the channel. For example:

```
/leave #hottub
```

When you enter a **/join** command to join a new channel, IRC will automatically remove you from your current channel. Once you are more experienced, you can issue the command:

```
/set novice off
```

This changes several things about your IRC session. In particular, it allows you to join more than one channel at the same time. (For more information, use **/help set novice**.)



FUN TIP: Once you use **/set novice off**, *you can join as many channels as you want. However, if you are trying to follow too many conversations at once, you will probably fry your brain.*

From time to time, you will see a message, called an *action*, that says that someone has done something. For example, you might see:

```
* Nipper is disgusted.
```

Actions are used to imply feelings and ideas where regular messages would be inadequate. To display an action, you use the **/me** command. Type **/me** followed by your action. Everyone on the channel will see your nickname, followed by the action. For example, if your nickname is **Nipper** and you enter:

```
/me breaks the glass.
```

Everyone (including you) will see:

```
* Nipper breaks the glass.
```

Nicknames

When you first connect to IRC, you can specify a nickname (which can be up to nine characters). At any time, you can change your nickname by typing **/nick**, followed by the new nickname. For example, if you nickname is **Irishboy** and you want to change it to **MikeP**, use:

```
/nick MikeP
```

When you do, IRC will notify everybody in your channel. You will all see a message like:

```
*** Irishboy is now known as MikeP
```

When you specify a nickname, you can type upper- or lowercase letters, and IRC will remember what you use. Whenever IRC displays your nickname, it will be just as you specified it. However, when you type a nickname as part of a command, you can use all lowercase letters, which is a lot easier to type.

When you specify a new nickname, IRC checks it against all the nicknames currently in use (ignoring any differences due to upper- and lowercase). If someone is already using the same name, you will be asked to select another one. If the nickname is registered to someone else, but is not currently in use, you will be allowed to use it, but you will see a warning. In such cases, the polite thing to do is to change to another nickname, so the person to whom the nickname is registered can use it if he or she connects.

If you enter the command by itself:

```
/nick
```

you will see your current nickname. If you would like to see the last person who used your current nickname, enter:

```
/whowas
```

To find out who was the last person to use a particular nickname, use **/whowas** followed by that name. For example:

```
/whowas nipper
```

IRC nicknames are maintained by an automated service called *Nickserv*. To learn how to use Nickserv to register your own nickname, enter:

```
/msg nickserv@service.de intro
```

This command sends a private message to the Nickserv program, asking it to display the introductory information.

Commands for Talking to Specific People

To send a private message to someone, use **/msg** followed by his or her nickname, followed by the message. For example:

```
/msg dizney What's up guy?
```

The message will be seen only on that person's screen.

When you send private messages, you do not have to be on the same channel as the recipient. Wherever the person is, he or she will get the message. (That is one reason why no two people can use the same nickname at the same time.)

As you look at your screen, normal messages are prefaced by the nickname of the originator, enclosed in < > characters. For example:

```
<Addie> Hi Nipper, how are you?
```

When someone sends you a private message, the nickname will be marked with * characters instead. For example:

```
*Addie* What are you working on?
```

When you send a private message to someone, it will show up on your screen with his or her nickname preceded by the characters **->**. For example, if you send a message to the person whose nickname is **Addie**, what you see on your screen looks like this:

```
-> *Addie* I am working on Chapter 20.
```

Hint

Because the world of IRC is so open and public, it is necessary to maintain a protective aura of anonymity. At any time, there may be thousands of people talking to one another, but each particular person is known only by a nickname. For this reason, many people consider it good manners to refer to a person only by his or her nickname, even if you know their real name.

When you use **/msg**, there are two shortcuts available. First, if you use a , (comma) character instead of a nickname, the message will go to the last person who sent you a private message. For example, if the last person who sent you a private message was **Nipper**, the following two commands would be equivalent:

```
/msg nipper What's up guy?
/msg , What's up guy?
```

Alternatively, if you use a . (period) character instead of a nickname, the message will go to the last person to whom you sent a private message. For example:

```
/msg . Did you get my last message?
```

If you want to send a private message to more than one person, you can specify more than one nickname. However, you must separate the names with a comma and not put in any spaces:

```
/msg nipper,addie  Hello you two.
```

Remember that any line that starts with a / (slash) character is interpreted as a command, and is not sent to the channel. Similarly, any line that does not begin with a / is considered to be part of the general conversation and is sent to the channel.

Thus, if you type a **/msg** command, but accidentally put a space in front of the / character (or omit the / character), the line would be broadcast for everyone to see.



FUN TIP: To make new friends, join a channel with a lot of people and then type the following line. Be sure to put a space in front of the / character.

```
 /msg . sounds great!  Shall I bring my whip with me too?
```

If you find yourself sending more than a few **/msg** lines to the same person, you can simplify things by using the **/query** command followed by one or more nicknames. As with **/msg**, you must separate the nicknames with a comma and not put in extra spaces.

This command tells IRC that, until further notice, everything you type (except commands, of course) should be sent to the specified people. Here are two examples:

```
/query addie
/query addie,nipper
```

The first command says that all our messages should go only to **addie**. The second command sends messages to both **addie** and **nipper**.

When you are finished with your private conversation, you can turn it off by using the command without a nickname:

```
/query
```

Now all your messages will go to your channel.

Hint

Even though you may be sending private messages with **/msg** or **/query**, your screen will still show all the other messages that are being sent to the channel, which can be distracting. If you want a quiet conversation with someone, ask the person to join you on another channel. For example, you might type:

```
/msg addie Let's talk more quietly. Join me on '#foobar'.
```

If you specify a channel name that is unused, you will have your own private channel. After you are both on the channel, you can keep it private by entering the command:

```
/mode * +pi
```

For more information about the **/mode** command, use **/help mode**.

Displaying Information About People

The last few commands that we will mention show you information about people. First, to display information about a particular person, use **/whois** followed by the nickname, for example:

```
/whois nipper
```

You also can display information about a person by using the **/who** command:

```
/who nipper
```

However, **/whois** is better as it shows you more information.

Where **/who** comes in handy is in showing you information about all the people in a particular channel. For example, to find out who has joined channel **#hottub**, use:

```
/who #hottub
```

If you want to see a list of everybody who is joined to your current channel, you can enter:

```
/who *
```

Hint

If you have not joined a channel, some IRC clients will interpret the command **/who** * as meaning that you want a list of everybody who is using IRC. Similarly, if you type **/whois** *, IRC will display information about everybody who is connected.

Do not do this, for the list will go on forever (or until you die, whichever comes first). If you do find yourself in this situation, remember that you can use **/flush** to stop the rest of the output (eventually).

Summary of Basic IRC Commands

For reference, Table 20-1 contains a summary of the basic IRC commands that we have covered in this chapter. These commands are enough to get you started. However, there are a lot more commands to learn and a lot of surprises waiting for you (like robots and the flower server).

Command	Description
`/flush`	throw away remaining output for current command
`/help`	display a list of all IRC commands
`/help` *`command`*	display help regarding specified command
`/help intro`	display an introduction to IRC
`/help newuser`	display information for new users
`/join` *`channel`*	join the specified channel
`/leave` *`channel`*	leave the specified channel
`/list`	display information about all channels
`/list` *`channel`*	display information about specified channel
`/list -min` *`n`*	display channels that have at least *n* people
`/list -max` *`n`*	display channels that have no more than *n* people
`/me` *`action`*	display specified action
`/mode * +pi`	make current channel completely private
`/msg` *`nicknames text`*	send private message to specified people
`/msg ,` *`text`*	send message to last person who sent you a message
`/msg .` *`text`*	send message to last person you sent a message to
`/nick`	display your current nickname
`/nick` *`nickname`*	change your nickname to specified name
`/query` *`nicknames`*	send all your messages to specified people
`/query`	stop sending private messages
`/quit`	quit IRC
`/set novice off`	allow certain actions (like joining multiple channels)
`/who` *`channel`*	show who is joined to specified channel
`/who` *`nickname`*	show information about the specified person
`/who *`	show who is joined to the current channel
`/whois` *`nickname`*	show all information about the specified person
`/whois *`	show all information about everybody

Table 20-1. *Summary of Basic IRC Commands*

CHAPTER 21

The Gopher, Veronica, and Jughead

If we could use only one Internet resource, it would be the Gopher. There are two reasons. First, the Gopher is easier to use than anything else on the Internet. Second, the Gopher allows you to access a larger variety of information and services than anything else on the Internet.

Does this mean that the Gopher is the ultimate Internet information tool? No, because there are many ways to think about information, and we need a variety of tools to serve a variety of people. We recommend that you learn to use all the basic Internet services so that, whatever your needs, you are always able to use the best tool for the job. For example, it is possible to

access Anonymous FTP files using the Gopher, but most of the time it is easier to use FTP directly.

Still, the Gopher is the closest thing we have to a one-size-fits-all service bureau and, as such, it provides a unique window into the Internet. Moreover, when you have spare time, the Gopher is probably the best tool for browsing for those unexpected and exotic items that make life on the Internet so interesting.

What Is the Gopher?

The Gopher is a powerful system that allows you to access many of the resources of the Internet in a simple, consistent manner. To use the Gopher, all you need to do is make selections from a menu. Each time you make a selection, the Gopher does whatever is necessary to carry out your request. For example, if you select a menu item that represents a text file, the Gopher will get that file—wherever it happens to be—and display it for you.

Some menu items represent other menus. If you choose one of these, the Gopher will obtain the new menu and display it for you. Thus, you can move from menu to menu, using only a handful of keys (or a mouse) to navigate.

The power of the Gopher lies in the fact that the resources listed in a menu may be anywhere on the Internet. When you select an item, the Gopher will get or do whatever is necessary to carry out that request. Much of the time, the Gopher will have to connect to another computer, but it will all be transparent to you. All you will notice is that your request has been fulfilled simply and easily.

How Does the Gopher Work?

The Gopher is a client/server system. (We discuss clients and servers in Chapter 2.) To use the Gopher, you run a program called a *Gopher client*. This program displays the menus for you and carries out your requests. From time to time, your Gopher client contacts a *Gopher server* to ask for information on your behalf. If it becomes necessary to contact another type of service—say, to set up a Telnet session, or to download a file—your Gopher client will take care of that as well.

There are several thousand Gopher servers around the Internet, and they store all manner of information. You will find Gopher servers in many universities, companies, and other organizations. Indeed, within a university, you will often find separate departments that have their own Gopher server.

Each Gopher server carries information that is of interest to the local users. For example, the Gopher server for a university department would have information that is of interest to the members of that department. The Gopher server at a company would have information for its customers and employees.

What is great about this system is that, for the most part, all the Gopher servers on the Internet are public. Although most Gopher servers are set up for a particular set of people, you will find that much of the information you encounter is of general interest.

Travelling Through Gopherspace

The Gopher offers more information than any single person can comprehend. We refer to the sum total of all this information—everything that is available via the Gopher—as *gopherspace*. To help you get a feeling for what travelling through gopherspace is like, here is a typical scenario that describes an everyday excursion.

In our example, you are a student at the Unix Studies Department at the University of Foobar. Your department, as well as others at the university, has its own Gopher server. To use the Gopher, you start your Gopher client program (we will explain how later). Your Gopher client connects to the local Gopher server and asks for the initial menu. The Gopher server sends this menu, which your Gopher client displays on your screen.

```
                Internet Gopher Information Client v1.30

      University of Foobar, Unix Studies Department Gopher Server

 -->  1.  About the Unix Studies Department.
      2.  Where to buy the book "A Student's Guide to Unix".
      3.  Search the Online Unix Manual <?>
      4.  The Internet Studies Department Gopher/
      5.  Gopher Servers at the University of Foobar/
      6.  Other Gopher Servers Around the World/
      7.  University of Foobar Directory <CSO>
      8.  The Unix Daemon <Picture>
      9.  University of Foobar Library Catalog <TEL>
     10.  The Sound of Unix <)
     11.  Fun and Games/

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

You decide that you want to know more about the Unix Studies department, so you select item number 1. Your Gopher client contacts the Gopher server and asks for the information. As you wait, you see the message:

```
Receiving Information..
```

A moment later, the Gopher server has sent back a short description of the department, which your Gopher client displays for you. After you are finished reading, you see the message:

```
Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:
```

You press RETURN, and your Gopher client re-displays the previous menu.

This time, you decide to see what another Gopher server might have to offer, so you select item number 6. Your Gopher client sends the request to the Gopher server, which sends back a new menu.

This menu contains a long list of many different Gopher servers. Your Gopher client displays the first screenful, which it calls a page. You move through the list, page by page. On the 37th page, you see the following:

```
660. West Virginia Network for Educational Telecomputing/
661. Western Illinois University, IL USA/
662. Wheaton College, Wheaton, IL/
663. Whole Earth 'Lectronic Magazine--The WELL/
664. Wisconsin Interlibrary Services/
665. Wittenberg University/
666. Worcester Foundation for Experimental Biology/
```

You decide to check out The Well, so you select item number 663. Your Gopher client connects to the Well's Gopher server—which happens to be in Sausalito, California, just north of San Francisco—and The Well's Gopher server sends its main menu to your Gopher client. Your Gopher client displays this menu. As you look at it, you notice the following item:

```
11. Publications (includes Zines)/
```

You happen to know that a "zine" is a small, personal, opinionated publication, devoted to a single group or topic, usually far from the center of mainstream culture. This item looks intriguing. You select it and your Gopher client sends the request to The Well's Gopher server, which responds by sending back another menu. Your Gopher client displays this menu. As you expected, you see a lot of weird-looking items, among which you notice:

```
10. Online Zines/
```

This looks even more intriguing: zines in electronic form that you can read right on your screen. You select this item and your Gopher client again passes the request to The Well's Gopher server. Once again, the Gopher server sends back a menu which your Gopher client displays. This menu contains a list of online zines. You select item number 16:

```
16. Obscure Electronic #4
```

Your Gopher client sends in the request. The Gopher server sends back its response: a copy of a zine called *Obscure Electronic* edited by Jim Romenesko of Milwaukee, Wisconsin. This issue consists of a long profile of Dan Kelly from Chicago, Illinois, who publishes his own zine called *EVIL: The Newsletter for True Crime Book Fanatics*. Your Gopher client displays the zine, one page at a time. As you read, you are fascinated to learn all about Kelly's long-time interest in serial killers. When you are finished, your Gopher client displays the message:

```
Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:
```

Since the article was so interesting, you decide to mail a copy to the President of the United States (his address is **president@whitehouse.gov**). You press the **m** key and your Gopher client asks you to enter a mail address. You enter the address and press RETURN. Your Gopher client mails a copy of the article as you requested. On second thought, you decide that you would like to have a copy for yourself, so you repeat the procedure and enter your own mail address.

You now move back up one level and display the previous menu (the one about online zines). This menu proved to be so useful that you use a special command to tell your Gopher client to remember the menu, so you can get to it quickly whenever you want.

Your Gopher client makes a note called a *bookmark*. Any time you want, you can ask your Gopher client to display your personal *bookmark list*. You can then select this item and have your Gopher client go to it directly, without having to make your way through so many servers and menus like you did the first time.

And now, feeling pleasantly exhausted from your travels through the idyllic islands of gopherspace, you bid a fond farewell to the gracious and friendly natives at The Well in Sausalito, California, and to the two wise men in Milwaukee and Chicago. As the sun sinks slowly over your gopherspace paradise, you press the **Q** (quit) key and stop your Gopher client.

Musing on the Gopher

As you will have noticed from the last section, your Gopher client allows you to move from one Gopher server to another, smoothly and easily. For this reason, we like to consider the hundreds of interconnected Gopher servers to be one large distributed entity; thus, we speak of "the Gopher", with its many small parts, spread throughout the Internet. The Gopher is always growing, always changing. It's everywhere, but you can't locate it; it's always there, but you can't see it.



FUN TIP: The Gopher is the largest and most practical example of applied pantheism in the history of mankind.

Thus, the existence of gopherspace depends upon two completely different facilities. First, there is the Gopher itself: a vast information-based lifeform consisting of many interconnected Gopher servers.

Second, there is your Gopher client: the program you use to visit gopherspace and to interact with the Gopher. For this reason, when we talk about "your Gopher", we mean your client program.

You might be wondering, where did the name "Gopher" come from? Certainly one can imagine an electronic version of a furry little creature who burrows through gopherspace, seeking out treasures on your behalf. Still, the analogy is not apt, for it is neither the Gopher nor your Gopher client that does the burrowing: it is your mind, jumping from topic to topic as you read the menus and make your choices. No, the actual origin of the name "Gopher" is more prosaic, but not without its points of interest.

The original Gopher was developed at the University of Minnesota in April of 1991 by a team consisting of Bob Alberti, Farhad Anklesaria, Paul Lindler, Mark McCahill, and Daniel Torrey. The work was done within the Department of Computer and Information Systems in order to provide a cheap and easy way for various campus departments to make information available to the campus at large. The idea was—and still is—that each interested organization can maintain their own Gopher server and put whatever they want on it. Thus, every organization has control over its own section of gopherspace, which it shares with anyone who has a Gopher client.

And what about the name "Gopher"? It happens that Minnesota is known as the Gopher State. For example, the sports teams at the main campus of the University of Minnesota are named the Golden Gophers. No one knows exactly why Minnesota is the Gopher State—the most widely accepted theory is that the

inventor of Minnesota (Phineas T. Bushbottom) actually resembled a rodent—still, there it is.

Moreover, there is a serendipitous association between the name "Gopher" and the American slang expression "gofer". This epithet describes a person whose position in life is not unlike that of a Gopher client: to go for this and go for that.

In the charming argot of the American midwest, "go for" becomes "gofer" which becomes "Gopher". Indeed, this is probably why so many tourists are astonished to find, upon visiting Minnesota, that the natives are so cooperative when it comes to running errands for other people.

Starting Your Gopher Client

There are a variety of Gopher clients available for different systems. The most widely used Gopher client is the one that runs on Unix systems with regular text-based terminals. The name of this program is **gopher**. There is also a Gopher client, named **xgopher**, for X Window systems. (We discuss X Window in Chapter 2.)

If you have a choice, use a Gopher client that takes advantage of the special features of your computer. For example, if your system uses pull-down menus, scroll bars, and a mouse, a Gopher client tailored to your computer will make use of these features.

If your system has a Gopher client installed, all you need to do is enter the name of the program to start it. For most Unix users, this program will probably be **gopher**. So to start your Gopher client, all you need to enter is:

```
gopher
```

If you are using an X Window system, you can certainly run **gopher**, but you may prefer **xgopher** if it is available.

You will remember that in the last section we explained that you can save locations in gopherspace to a menu of your own. Each location you save is called a bookmark. If you would like to start your Gopher client with the menu that holds your personal bookmarks—rather than with the main menu of the default Gopher server—specify the **-b** (bookmark) option when you start the program:

```
gopher -b
```

From time to time, you may have the address of a particular Gopher server that you want to use. If so, you can enter the address as part of the **gopher** command. This tells your Gopher to connect directly to the specified server instead of using whatever Gopher server is the default.

For example, in the catalog in the back of this book, there is a Gopher at Northwestern University that specializes in aviation. The address of the Gopher is **av.eecs.nwu.edu**. If you want to connect directly to this particular Gopher server, you can enter:

```
gopher av.eecs.nwu.edu
```

Some Gopher servers require that you specify a particular port number (see Chapter 7). For example, the catalog mentions a Gopher at the University of Tuebingen in Germany that contains documents about MUD (Multiple User Dimension role-playing games). The address is listed as **nova.tat.physik.uni-tuebingen.de 4242**. The **4242** is the port number. Simply type it as part of the command:

```
gopher nova.tat.physik.uni-tuebingen.de 4242
```

If you enter the **gopher** command and you see the following message:

```
gopher: Command not found.
```

it means that your system does not have a Gopher client installed. You have several choices, which we will discuss in the next section.

If you are using a Unix system and you do have a Gopher client, you can display the official documentation about the **gopher** command by using the **man** command to access the online Unix manual:

```
man gopher
```

(Note: This will display documentation about the **gopher** command, not about the Gopher system in general.) You can also use **man** to display documentation about the **xgopher** command, if it is available on your system:

```
man xgopher
```

Once you start your Gopher, it will contact whatever Gopher server is designated as the place to start and ask for the main menu. If your organization

has its own Gopher server, this, no doubt, will be where you will start. Otherwise, your system manager will have chosen some other Gopher server, somewhere on the Internet, as the starting place. Once you see the main menu, you are ready to begin: simply make your first selection.

If you want to learn more about the Gopher, there are several places to look. First, many Gopher servers have menu items that display information about the Gopher itself.

Second, there are two Usenet newsgroups devoted to Gopher-related discussions: **comp.infosystems.gopher** and **alt.gopher**.

Finally, you can read the Gopher frequently asked question (FAQ) list. You can obtain this document via Anonymous FTP from **rtfm.mit.edu**. Change to the directory **/pub/usenet/news.answers** and download the file named **gopher-faq**. (We discuss Usenet newsgroups and FAQ lists in Chapter 9. We discuss Anonymous FTP in Chapter 16.)

Public Gopher Clients

If your system does not have a Gopher client installed, you have two choices. In the long run, the best solution is to install a Gopher client. Ask your system manager if he or she would be so kind as to download the program via Anonymous FTP and install it for everybody. The location of where to find such clients is given in the Gopher FAQ list.

The second choice is to use a public Gopher client. All you need to do is telnet to one of the hosts in Table 21-1. Once you log in using the specified userid, a Gopher client will start automatically. (We discuss Telnet in Chapter 7.)

Hint

Using a public Gopher client will somewhat restrict what you can do. For example, after your Gopher has displayed a text file, you will see the following message:

```
Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:
```

If you are using a Gopher client on your own computer, you can use all three options. However, if you have telneted to a remote Gopher client, you can only use mail (because you cannot save or print files on a remote computer).

You may also find that, as a remote user, you are denied other privileges, such as being able to initiate a Telnet session.

Location	Internet Address	IP Address	Log in as...
Australia	info.anu.edu.au	150.203.84.20	info
Chile	gopher.puc.cl	146.155.1.16	gopher
Denmark	gopher.denet.dk	129.142.6.66	gopher
Ecuador	ecnet.ec	157.100.45.2	gopher
England	gopher.brad.ac.uk	143.53.2.5	info
Germany	gopher.th-darmstadt.de	130.83.55.75	gopher
Japan	gopher.ncc.go.jp	160.190.10.1	gopher
Spain	gopher.uv.es	147.156.1.12	gopher
Sweden	gopher.chalmers.se	129.16.221.40	gopher
Sweden	gopher.sunet.se	192.36.125.2	gopher
USA: California	infopath.ucsd.edu	132.239.50.100	infopath
USA: California	scilibx.ucsc.edu	128.114.143.4	gopher
USA: Georgia	grits.valdosta.peachnet.edu	131.144.8.206	gopher
USA: Illinois	gopher.uiuc.edu	128.174.5.61	gopher
USA: Iowa	panda.uiowa.edu	128.255.40.201	–
USA: Michigan	gopher.msu.edu	35.8.2.61	gopher
USA: Minnesota	consultant.micro.umn.edu	134.84.132.4	gopher
USA: North Carolina	gopher.unc.edu	152.2.22.81	gopher
USA: North Carolina	twosocks.ces.ncsu.edu	152.1.45.21	gopher
USA: Ohio	gopher.ohiolink.edu	130.108.120.25	gopher
USA: Virginia	ecosys.drdr.virginia.edu	128.143.96.10	gopher
USA: Virginia	gopher.virginia.edu	128.143.22.36	gwis
USA: Washington	wsuaix.csc.wsu.edu	134.121.1.40	wsuinfo

Table 21-1. *Public Gopher Clients to Which You Can Telnet*

Basic Gopher Commands

There are a variety of commands that you can use to negotiate gopherspace. Most of the time, you can go wherever you want by using just six keys: the four cursor control keys (with the arrows) RIGHT, LEFT, DOWN, and UP; the SPACE key; and the **B** key. We will go into the details in a moment.

There are three basic commands that you should always remember. First, to display help information about all the commands, press **?** (question mark).

Second, to stop your Gopher client, press **q** (quit). You will be asked if you really want to quit, to which you can answer **y** (yes) or **n** (no).

Finally, to quit immediately, without being asked for confirmation, press **Q** (uppercase "Q").

Your Gopher works in what is called *cbreak mode*, which means that, for single character commands, you do not have to press the RETURN key. For example, to stop your Gopher, you need only press the **q** key. Do not press RETURN.

Now, let's take a look at a typical Gopher menu:

```
                Internet Gopher Information Client v1.30

    University of Foobar, Unix Studies Department Gopher Server

--> 1.  About the Unix Studies Department.
    2.  Where to buy the book "A Student's Guide to Unix".
    3.  Search the Online Unix Manual <?>
    4.  The Internet Studies Department Gopher/
    5.  Gopher Servers at the University of Foobar/
    6.  Other Gopher Servers Around the World/
    7.  University of Foobar Directory <CSO>
    8.  The Unix Daemon <Picture>
    9.  University of Foobar Library Catalog <TEL>
   10.  The Sound of Unix <)
   11.  Fun and Games/

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

At all times a pointer, shown as -->, will be beside one of the items. In this case, it is pointing to item number 1. The idea is to move the pointer to the item you want and then select it. To select an item, you can press either RIGHT or RETURN.

If you select another menu, your Gopher will get it for you and display it. If you select a text file, your Gopher will get a copy and display it one page at a time. For other types of menu items, your Gopher will do whatever is appropriate. (We will have more to say about this later in the chapter.)

Moving Around a Gopher Menu

As we mentioned, your current position within a menu will be marked by a pointer. There are several ways to move this pointer. The easiest way is to use the UP and DOWN keys. If your keyboard does not have these keys, or if you prefer not to use them, you can use **k** or CTRL-P to move up, and **j** or CTRL-N to move down. Although these choices may seem odd, they are used within the **vi** text editor and are second nature to a great many Unix users.

If you would like to jump to and select a particular item, simply enter its number. For example, to select item number 10, type **10** and then press RETURN.

Some menus are so long that they will not fit onto a single page. In this case, the message at the bottom right will tell you that there is more than one page. For example, you might see:

```
Press ? for Help, q to Quit, u to go up a menu          Page: 1/7
```

To move to the next page, press SPACE. To move to the previous page, press **b** (back). Again, there are alternative keys to use. To move to the next page, you can press > (greater than) or + (plus). To move to the previous page, you can press < (less than) or – (minus).

If a menu is so long as to require more than one page, you can jump to and select an item, even if it is not on the current page. For example, say that you are looking at page 7 of a long menu. You can select item number 1 directly. Simply type **1** and press RETURN.

Another way to jump is to tell your Gopher to search the current menu for an item that contains a particular pattern. To do so, type a / (slash) character, followed by the pattern, and then press RETURN. When it searches, your Gopher does not distinguish between either upper- or lowercase letters.

For example, say you are looking at a very long list of Gopher servers, and you would like to jump directly to the item for the University of Foobar. Enter:

```
/foobar
```

If the item your Gopher finds is not the one you wanted, you can repeat the same search by pressing the **n** (next) key.

As you know, many of the items you select will themselves be menus. When you select such an item, your Gopher will request the menu and then display it for you. Thus, travelling through gopherspace entails moving up and down a tree of menus.

There are a few commands that you can use to move from one menu to the next. As we mentioned above, to select a new menu, point to that item and press RIGHT or RETURN. To return to the previous menu, press LEFT or **u** (for up).

Now you can see the beauty of the system. Although there are many Gopher commands, all you really need to navigate gopherspace are UP and DOWN (to move within a menu), and RIGHT and LEFT (to move from one menu to another). To this basic command set, you can add SPACE and **b** to page forward and backward through long menus.

At any time, you can press **m** to jump to the main menu (the one your Gopher started with). This is handy when you are far down a series of menus and you want to jump back to the beginning.

Saving a File

There are several ways to save a file. As we mentioned earlier, once your Gopher has displayed a file, you will see:

```
Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:
```

You can now mail a copy of the file to someone, save it, or print it.

There is a shortcut for saving a file that does not require you to display it first. Simply move to the menu item and press **s** (save). You will then be asked to supply a file name. Your Gopher will suggest a file name that you can accept or modify as you wish. For example, say that the current menu item is:

```
  --> 16. Obscure Electronic #4
```

You decide to save it to a file. To do so, you press **s**. You now see:

```
+-------------------------------------------------------------+
|                                                             |
| Save in file:   Obscure-Electronic-#4                       |
|                                                             |
|                        [Cancel ^G] [Accept - Enter]         |
|                                                             |
+-------------------------------------------------------------+
```

You can now either press RETURN to save the file, or press CTRL-G to forget the whole thing. Before you save, you can change the name by typing a new one.

Saving has two limitations. First, the current item must be something that would be meaningful if it were saved in a personal file. For example, you can save a text file, but you cannot save a menu. Second, you must be using a Gopher client that is running on your own computer. This makes sense as you do not have permission to save files on a remote host.

Another way to save a file is by downloading. This is handy when you are using a PC or Macintosh to connect to a remote Internet host. You can use the **D** (uppercase "D") command to download an item directly to your computer. (We discuss the basic PC-to-host connection in Chapter 3. We discuss downloading in Chapter 18.)

When you press **D**, you will see a menu asking you to choose which file transfer protocol (download method) you want to use:

```
+----------Obscure Electronic #4-------------+
|                                             |
| 1. Zmodem                                   |
| 2. Ymodem                                   |
| 3. Xmodem-1K                                |
| 4. Xmodem-CRC                               |
| 5. Kermit                                   |
| 6. Text                                     |
|                                             |
| Choose a download method:                   |
|                                             |
| [Cancel ^G]  [Choose 1-6]                   |
|                                             |
+---------------------------------------------+
```

After you make a choice, your Gopher will initiate the file transfer. Depending on how your communication program is set up, you may then see a message telling you to start your download. (The whole thing works just like we described in Chapter 18.)

Customizing Your Gopher Environment

As we explained earlier, you can use a bookmark to save a menu item. Your Gopher stores all your bookmarks in a bookmark list that you can jump to whenever you want. This allows you to remember and visit any location in gopherspace. If you are using a Gopher client on your own machine, your bookmark list will be saved automatically, so you can use it each time you explore gopherspace.

There are several commands that you can use to create your personal bookmark list. To add a menu item to your list, move to it and press **a** (add). To add the entire menu to your bookmark list, press **A** (uppercase "A").

At any time, you can jump to your bookmark list (which is itself a menu) by pressing **v** (view). To return to the previous menu, simply press LEFT or **u**.

While you are looking at your bookmark list, you can delete an item by moving to it and pressing **d** (delete).

Hint

The art of using the Gopher well is to build up a personalized bookmark list. Many people forget to use this facility and end up retracing their steps time and time again. Moreover, it is all too easy to forget where in gopherspace you found a particular item. If you do not save interesting items to your bookmark list, you may never see them again.

The next command displays technical information about any menu item. Simply move to an item and press = (equal sign). You will see the technical description of the item as it appears in the Gopher server's database.

For example, say that you are pointing to the following item:

```
--> 1.  CIA World Fact Book/
```

If you press =, you will see something similar to:

```
Type=1
Name=CIA World Fact Book
Path=1/info/Government/Factbook
Host=info.umd.edu
Port=901
```

You don't really need to worry about the details. What is interesting, though, is that you can see where in gopherspace the item is located. In this case, the item is stored on **info.umd.edu**, a computer at the University of Maryland in the United States.

The last command that we will mention is **O** (uppercase "O"). This tells your Gopher to display a list of options. You can examine the list and make changes as you wish. Most of the time, you will probably not want to make any changes, so if you don't understand what something does, you are best off leaving it alone.

Summary of Gopher Commands

For reference, Figure 21-1 contains a summary of the Gopher commands that we discussed. Remember, you can display a command summary at any time by pressing ?.

Basic Commands

`Q`	quit Gopher immediately
`q`	quit Gopher, but ask for confirmation
`?`	display a help summary
`=`	display technical information about an item
`O`	examine and change Gopher options

Fundamental Commands to Move Through Gopherspace

RIGHT	select the current item
LEFT	back up one level to the previous menu
UP	move pointer up one item
DOWN	move pointer down one item
SPACE	move to next page of the menu
`b`	move back to previous page of the menu
`number`	jump to and select specified item
/pattern	search for next menu item containing pattern
`n`	search for next menu item using same pattern
`m`	jump to the main menu

Alternate Commands for Moving the Pointer

RETURN	select the current item
`u`	back up one level to the previous menu
`k`	move pointer up one item
`j`	move pointer down one item
CTRL-P	move pointer up one item (previous)
CTRL-N	move pointer down one item (next)
`>`	move to next page of the menu
`<`	move to previous page of the menu
`+`	move to next page of the menu
`-`	move to previous page of the menu

Saving Information

`s`	save current item to a file
`D`	download current item to a file

Using Bookmarks

`a`	add current item to the bookmark list
`A`	add current menu or search to the bookmark list
`d`	delete a bookmark
`v`	jump to the bookmark list (view)

Figure 21-1. *Summary of Gopher commands*

Types of Gopher Resources

Let's take another look at a typical Gopher menu. Notice that each item ends with a symbol. The symbol tells you what type of resource is represented. In this section, we will discuss the various resources that you will find and explain how to use them.

```
                Internet Gopher Information Client v1.30

      University of Foobar, Unix Studies Department Gopher Server

 --> 1.  About the Unix Studies Department.
     2.  Where to buy the book "A Student's Guide to Unix".
     3.  Search the Online Unix Manual <?>
     4.  The Internet Studies Department Gopher/
     5.  Gopher Servers at the University of Foobar/
     6.  Other Gopher Servers Around the World/
     7.  University of Foobar Directory <CSO>
     8.  The Unix Daemon <Picture>
     9.  University of Foobar Library Catalog <TEL>
    10.  The Sound of Unix <)
    11.  Fun and Games/

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

The most common symbol that you will see at the end of an item is a / (slash) character. This indicates that the item represents another menu. For example, look at item number 4.

```
     4.  The Internet Studies Department Gopher/
```

If you choose this item, you will get the main menu for the Internet Studies Department Gopher server.

One thing you may have noticed is that the Gopher menu structure resembles the Unix file system. Menus and sub-menus correspond to directories and sub-directories. For this reason, you will sometimes see Gopher menus referred to as directories. For example, when you are waiting for a menu to be fetched from a remote Gopher server, you will see the message:

```
Retrieving Directory...
```

(We discuss the Unix file system in Chapter 16.)

Hint

From time to time, you will notice that the Gopher is slow. For example, you may have to wait while your Gopher client connects to a remote Gopher server. Or you may initiate a database search (see below) that seems to go on and on.

In such cases, there is not much that you can do except wait. If you are a Unix user, you may be tempted to press CTRL-C, the key that is usually used to abort something. Unfortunately, all that will happen is that you will see a message asking if you want to abort the Gopher client itself. There is no simple way to stop a Gopher activity that is in progress.

The next most common symbol is a . (period) character. This indicates that the item is a text file: something to be displayed. For example, item number 1 represents a text file that contains information about the Unix Studies Department:

```
1.  About the Unix Studies Department.
```

When you select such an item, your Gopher will fetch it and display it for you, one page at a time.

Hint

If you are a Unix user using your own Gopher client, you can specify the paging program you prefer to use. To do so, set the **PAGER** environment variable to the name of your favorite paging program. For example, if you use the C-Shell, you could place the following command in your **.login** initialization file:

```
setenv PAGER more
```

If you use the Korn shell or Bourne shell, you could place the following two commands in your **.profile** initialization file:

```
PAGER=more
export PAGER
```

This will ensure that the **PAGER** environment variable is defined properly each time you log in. Moreover, this variable will be used by a number of programs (such as your mail program) and not only by your Gopher client.

The discussion of environment variables and initialization files is beyond the scope of this book. For more information, see a good Unix book. We recommend *A Student's Guide to Unix*, by Harley Hahn (McGraw-Hill).

The <TEL> symbol indicates a completely different type of resource: a Telnet session. In our example, we see:

```
9.  University of Foobar Library Catalog <TEL>
```

If we select this item, our Gopher will initiate a Telnet connection to a remote host. In this case, it is a host that offers access to the University of Foobar library catalog. Just before the connection is made, you will see a warning telling you that, in order to make the connection, you are going to have to go outside the Gopher system:

```
+---------University of Foobar Library Catalog-----------+
|                                                         |
| Warning!!!!!, you are about to leave the Internet       |
| Gopher program and connect to another host. If          |
| you get stuck press the control key and the ] key,      |
| and then type quit                                      |
|                                                         |
|                          [Cancel - ^G] [OK - Enter]     |
|                                                         |
+---------------------------------------------------------+
```

What this message is telling you is that you are about to embark on a regular Telnet connection. To continue, press RETURN. To forget the whole thing, press CTRL-G.

Once you finish with the Telnet connection, you will be sent back to your Gopher client. However, during the session, whatever you type will be under the auspices of Telnet. In particular, you can use the Telnet escape key, CTRL-]. For more details on how to conduct a Telnet session and a discussion of the Telnet commands, see Chapter 7.

The next Gopher symbol is **<CSO>**. This indicates a facility called a *CSO name server* that contains information about the people at a particular organization. For example:

```
7.  University of Foobar Directory <CSO>
```

In this case, you can select this item to search for information about someone at the University of Foobar. For example, you may need to find out someone's electronic mail address.

The name "CSO" refers to the Computing Services Office at the University of Illinois, Urbana, where the software was first developed. There are many CSO

name servers on the Internet, and it is common to see one on the main Gopher menu of a university or other large organization. CSO name servers belong to a family of programs called "white pages directories" that are used to find someone on the Internet. (We discuss such services in Chapter 22.)

If you select a **<CSO>** item, you will see a screen asking you what to search for. The following example shows one such screen. If you use a number of CSO name servers, you will encounter different variations. However, in principle, they all work the same way.

```
+---------------University of Foobar Directory---------------+
|                                                             |
|name       _____________________________                     |
|email      _____________________________                     |
|department _____________________________                     |
|                                                             |
| [Switch Fields - TAB]          [Cancel ^G] [Accept - Enter] |
|                                                             |
+-------------------------------------------------------------+
```

Each of the categories—**name**, **email**, and **department**—is called a *field*. All you have to do is fill in whatever information you know and press RETURN. The CSO server will check its database and display what it finds as a single text file.

As you are typing, you can move from one field to another by pressing the TAB key. If you decide to forget about searching, simply press CTRL-G to return to the last Gopher menu.

The next symbol to discuss is **<?>**. This indicates a database that you can search by entering one or more keywords. For example:

```
3.  Search the Online Unix Manual <?>
```

When you select this item, you will see a screen that asks you to specify what you want to search for. For example:

```
+---------------Search the Online Unix Manual----------------+
|                                                             |
|Words to search for _____________________                    |
|                                                             |
|                        [Cancel ^G] [Accept - Enter]         |
|                                                             |
+-------------------------------------------------------------+
```

You fill in one or more keywords and press RETURN. Your Gopher will initiate the search. Once the results come back, your Gopher will display them for you in the form of a menu. Simply select whatever looks good.

Of course, the results of your search are only as good as the database you are searching and only as exact as the keywords you specify. One problem is that, unless the description of the item tells you, you have no way to know what type of search program and database you are using. If you are looking for files, you may be using Archie (Chapter 17). If you are searching plain text, you may be using Wais (Chapter 23).

Hint

You can use the = command to display technical information about a <?> menu item. This will sometimes give you a hint about what type of database is being searched. For example, you may see the word "wais" within the information.

The best thing to do is experiment. If you enter a keyword that doesn't get you what you want, try something else. Remember that Archie can only search for a single word, but Wais will search for more than one word.

Hint

It is often better to use Wais directly than through a Gopher client. This is because the Gopher client limits you as to what types of searches you can perform. When you use Wais directly, you can take advantage of all the Wais resources. For more information about Wais and how to use it, see Chapter 23.

If you like how a search turned out, you may want to save it to use again on another day. To do so, wait until the search is finished and use the **A** command. This will save the search (not its results) to your bookmark list. Whenever you want, you can jump to your bookmark list and select this exact search.

For example, some Gopher servers allow you to search news articles that arrive via Clarinet (see Chapter 9). You can create a specific search that checks the news for your favorite topic, and then save the search to your bookmark list. Now, each day when you start work, you can call up your bookmark list and use your preconfigured search to look for new articles.

The last two menu symbols we will discuss are **<Picture>** and **<)**. The **<Picture>** symbol indicates a binary file that contains a picture. The **<)** symbol indicates a binary file that contains a sound. (The **<)** is supposed to look like a stereo speaker.) Here are two examples showing these symbols:

```
 8.  The Unix Daemon <Picture>
10.  The Sound of Unix <)
```

You can only access such items if your computer has the appropriate hardware and software. If this is the case, your system manager will have specified which programs to use at the time your Gopher client was installed. All you have to do is select the item.

Hint

If you are connecting to an Internet host by using a PC or Macintosh that is able to display pictures or play sounds, you can use the **D** command (described earlier) to download such files to your own computer. Of course, your computer must be able to understand the particular file format that is being used. For example, you can't play sound files from a Next machine on your PC.

Using Veronica to Search Gopherspace

Veronica is a Gopher-based resource that you can use to search gopherspace for all the menu items that contain specified words.

For example, say that you are interested in math jokes. You know that somewhere out in gopherspace, there must be some menu items that contain math jokes. But where are they? Veronica can find out.

Accessing Veronica is easy. It is simply a menu item that you can select like any other. Look for an item similar to the following:

```
 2.  Search titles in Gopherspace using veronica/
```

You will often find Veronica under the same heading as a list of Gopher servers. For example, a good place to look would be within an item like:

```
 8.  Other Gopher and Information Servers/
```

Hint

You only need to find Veronica once. Once you do, you can place it in your personal bookmark list by using the **a** (add) command. At any time, you can use the **v** (view) command to display this list and then select Veronica.

Once you select a menu item that mentions Veronica, you will see a list of Veronica-oriented items. Here is a typical example:

```
                Search titles in Gopherspace using veronica

--> 1.  Search gopherspace using veronica at NYSERNet <?>
    2.  Search gopherspace using veronica at University of Pisa <?>
    3.  Search gopherspace for GOPHER DIRECTORIES  (NYSERNet) <?>
    4.  Search gopherspace for GOPHER DIRECTORIES  (U. Pisa) <?>
    5.  How to compose veronica queries.
    6.  FAQ:  Frequently-Asked Questions about veronica.
```

In this example, you would choose item 1 or 2 to make a general search of gopherspace. (It is usually best to choose the location that is closest to you.) Veronica will ask you to specify what you are looking for:

```
+--------Search gopherspace using veronica at NYSERNet---------+
|                                                               |
|Words to search for _____________________                      |
|                                                               |
|                          [Cancel ^G] [Accept - Enter]         |
|                                                               |
+---------------------------------------------------------------+
```

We happen to be looking for math jokes, so we type **math** and press RETURN.

Once you press RETURN, Veronica will search all of known gopherspace for menu items that contain the word math. These items will be selected and placed in a menu of their own which will be presented to you. Here is an example of a few such items:

```
--> 1.  sci-math-faq.
    2.  space_math.
    3.  AMATH1.ZIP - Animated Math: counting/addition/subtraction.
    4.  UCALC21.ZIP math expression evaluator.
```

The nice thing about Veronica is that it shows you its results in the same form as a regular Gopher menu. Thus, you can use the standard Gopher commands to

examine the results of a Veronica search. If you like what you see, you can use the **a** command to add an item to your personal bookmark list.

When Veronica performs a search, it does not distinguish between upper- and lowercase letters. Thus, you get the same results searching for **math**, **Math**, or **MATH**. Indeed, you will notice that item number 3 above contains the word **Math**.

In our example, we are looking for math jokes, but our search was too general. Veronica has found all kinds of items that have to do with math, but are not funny. The best idea is to perform another search, this time looking for two words:

```
math joke
```

This search yields better results. Here are the first few items:

```
--> 1.  misc.math.jokes.
    2.  sci.math #6242 - Re: Math Jokes Needed
    3.  sci.math #6272 - Re: Math Jokes Needed
    4.  Re: Math jokes wanted ....
```

Evidently, we have found a number of Usenet articles (see Chapters 9 and 10). We are getting closer, but it would be nice to narrow down the search criteria even further.

To do so, Veronica allows us to specify certain qualifiers: **and**, **or**, and **not**. We can also group words in parentheses. Here are some examples.

When you use **and**, it tells Veronica to search for items that contain more than one word. By default, Veronica considers that you are using **and** whenever you specify more than one word. For example, the following two search lists are equivalent:

```
math jokes
math and jokes
```

You can use **or** to tell Veronica to search for items that contain any of the specified words. For example, to find all the items that contain either the word **jokes** or the word **humor** (or both), search for:

```
jokes or humor
```

You can use parentheses to indicate that a particular combination should be treated as a single word. For example, to find all the items that contain the word **math** and at least one of the words **jokes** or **humor**, search for:

```
math and (jokes or humor)
```

As we mentioned, Veronica will assume that you are using **and** by default. Thus, the last search is equivalent to:

```
math (jokes or humor)
```

Finally, you can use **not** in indicate that the item must not contain a specific word. For example, to find all the items that contain the words **math** and **jokes**, but not the word **wanted**, you would search for:

```
math jokes not wanted
```

There will be times when you want to search for (or not search for) words that are similar to one another. For example, you may want to search for items that contain **mathematics** as well as **math**, or **joke** as well as **jokes**. To do so, you can use a * (asterisk) character at the end of a word to represent any number of extra characters.

For example, to find all the items that contain any word that begins with **math** and any word that begins with **joke**, you would search for:

```
math* joke*
```

To find all the items that contain these same words but do not contain any word that begins with either **want** or **need**, search for:

```
math* joke* not (want* or need*)
```

The last thing that we will mention about searching is to remind you that we had two choices as to what type of Veronica search we wanted to use:

```
1.  Search gopherspace using veronica at NYSERNet <?>
3.  Search gopherspace for GOPHER DIRECTORIES  (NYSERNet) <?>
```

The regular search (number 1 in our example) finds all kinds of menu items. The directory search (number 3) only finds items that are themselves menus. If you are looking for something that you know is a menu, use a directory search. Not only will it be faster, but the results will contain fewer irrelevant items.

Hint

You may find that a Veronica search comes up with nothing when you know there must be something. For example, if you search for **math*** and Veronica cannot find anything, you should be suspicious.

There are several reasons why this may happen. The most common reason is that you have specified a bad search pattern. Ask Veronica to search again and, this time, re-check your spelling and your logic.

The second most common reason is that the remote Veronica host may not be working. If you suspect this is the case, try using a different Veronica host or try again later.

Learning More About Veronica

To learn more about Veronica, there are two places to look, both of which are accessible from most Veronica menus. First, you can read the article on "How to compose Veronica queries". This will explain all the details of how to tell Veronica what you want.

Second, you can read the Veronica frequently asked question list. This contains many questions and answers about Veronica.

To satisfy your curiosity, we will answer one interesting question right now: how does Veronica work?

The first version of Veronica was developed in November 1992. The work was done by Steven Foster and Fred Barrie of the University of Nevada at Reno, System Computing Department. Foster and Barrie were Gopher users who realized that the potential of the Gopher could never be reached unless there was an easy way to find items in gopherspace. They saw how frustrating and unproductive it was to hunt for specific items when you didn't know where to look.

For inspiration, they turned to Archie, which had been developed to provide a searchable database for the Internet's Anonymous FTP archives. They developed what they called a *Veronica server* to provide a similar function for the Gopher.

On a regular basis (every one to two weeks), their Veronica server would contact every known Gopher server and ask for a copy of all the menus. These

menus were stored in a database. At any time, a user could use a Veronica search program to look for items in the database.

It wasn't long before Veronica became an integral part of the Gopher culture. Today, there are a number of Veronica servers around the Internet, and new work on Veronica is coordinated with mainstream Gopher development.

Of course, Veronica is not perfect. It is too easy to come up with a lot of irrelevant items or even no items at all. Some of this has to do with how Veronica works; some of this reflects the nature of gopherspace itself. After all, each Gopher server is maintained locally and there are no global standards as to how menu items should be named or organized. Nor should there be.

The strength of the Gopher—and the Internet as a whole—lies in the fact that no one person or organization is in charge. This makes for a fertile environment in which tools such as Anonymous FTP, Archie, the Gopher, and Veronica can develop. Of course, it also makes for a certain amount of disorganization and chaos.

Using Jughead to Search Gopherspace

As you know, Veronica is a tool that allows you to search all of gopherspace. Jughead is a similar tool, except that it searches only a confined area of gopherspace. Jughead was developed by Rhett (Jonzy) Jones at the University of Utah Computer Center and was first released on March 25, 1993.

The reason that Jughead is so important is that there are times when you will want to search only one small area of gopherspace. For example, if you are a user at a university, you may want to search only those Gopher servers within the university. Similarly, if you use a public access Internet service that maintains a large Gopher server, it is handy to be able to search only the menu items on that server.

The same thing applies to outside Gopher users. If you are interested in finding something, say, at a particular university, you would like to be able to connect to one of their Gopher servers and search all the gopherspace at the university.

When a system manager decides that his users would benefit from some type of limited gopherspace search, he sets up a *Jughead server*. The role of the Jughead server is to maintain a database of all the menu items within the part of gopherspace specified by the system manager. At any time, a user can use Jughead to search for whatever he wants within that limited area.

For example, let's say that someone is interested in books and libraries at a particular university. He might think about using Veronica to search for **book* librar***. However, there are three problems with this approach. First, it would find menu items throughout all of gopherspace, which would be too much. Second, Veronica has a built-in limit as to how many items it will find on a single

search, and this limit may be exceeded before Veronica finds anything at the particular university. Third, it will take a while for Veronica to perform such a general search.

However, if the person uses Jughead, he can confine his search to the university. Moreover, the search will be faster than a Veronica search and will yield complete results. The only limitation is that someone must have already set up a Jughead server for that part of gopherspace.

How do you access Jughead? The same way you access Veronica: by selecting an item on a Gopher menu. The difference is that, where there are only a relatively few Veronica servers in the world, there are many Jugheads, each offering a service for a particular part of gopherspace. Here are some typical Jughead menu items:

```
4.  Use Jughead to search menus of University of Utah <?>
5.  Use Jughead to search menus of State of Utah <?>
```

Performing a Jughead search is a lot like searching with Veronica. You enter one or more words and Jughead looks for them. The results are returned to you as Gopher menu items, so you can select them in the regular manner. When you compose a Jughead search, you can use the special words **and**, **or**, and **not**. (For more information and some examples, see the previous discussion about Veronica.)

If you want to display help, all you need to do is use Jughead to search for **?** (question mark). You will then see an item for Jughead help, which you can display.

The Names Veronica and Jughead

The name Archie (Chapter 17) was used for the Anonymous FTP searching tool because it sounds like the word "archive". As Archie became popular, it became common to refer to it as if it were a person.

When Steven Foster and Fred Barrie needed a name for their Gopher search tool, they thought immediately of Archie Andrews: the perennial teenager who is the mainstay of a whole family of American comic books, comic strips, and cartoons.

In the comics, Archie has several friends, including Veronica and Jughead. Foster and Barrie chose Veronica. Later, Rhett (Jonzy) Jones chose Jughead.

If you are an Archie aficionado, you will know that Archie has other well-known friends, among them Betty, Reggie, and Moose. You might ask, will we soon see their names on the Internet? We think so.

Hint

As you now know, the names Veronica and Jughead were chosen because of their affinity to the Archie Andrews comics. Thus, there is no truth to the rumor that the name Veronica really stands for "Very Easy Rodent-Oriented Netwide Index to Computerized Archives". Nor does the name Jughead really mean "Jonzy's Universal Gopher Hierarchy Excavation And Display".

You will see these long names, but they were contrived after the fact. This may be disappointing to those people who love cryptic acronyms, but as one of our readers, we thought you deserve to know the real truth.



FUN TIP: If you are at all inclined to metaphysical speculation, think about this: in the comics, Jughead's last name is Jones. On the Internet, the Jughead program was developed by Rhett Jones. Cosmic or what?

CHAPTER 22

Finding Someone on the Internet

If you know someone's name, how do you find his or her Internet address? When you see a userid and address in a Usenet article, is there a way to find out who that person is? Or let's say you have a friend at a particular university, how do you send him mail?

These are some of the most common questions that Internet users ask. There are many different situations and variations, but it all boils down to the same thing: How do you find someone on the Internet?

In this chapter, we will explain what Internet facilities exist for locating people and show you how to use them. We will also show you how to find the location of a particular computer, when all you know is its name.

An Approach to Finding Someone

As you know, there is no central authority to organize the Internet. Indeed, there are so many computers on the Internet that no one can even keep track of all of them. Thus, it is not surprising that there is no such thing as a central Internet users directory.

This means that if you are looking for someone on the Internet, there is no one place to look. What you need to do is develop a strategy.

In this chapter, we will show you what facilities exist for finding people. We suggest that you become familiar with the entire chapter. Then, when you need to track down someone, you can decide where is the best place to start.

Although there is no single Internet directory, there are many services that will help you find someone. We can divide these services into three categories. If you know the organization to which the person belongs, you can try to find a white pages directory that contains information about the people in that organization. If the person sends articles to Usenet, you can use the Usenet address server. Or, if you have some idea where the person is located, you can use Netfind.

Now, before we start, here is a general hint that pertains to all the services we will discuss in this chapter.

Hint

In general, programs that search for names do not distinguish between upper- and lowercase letters. For example, if you want to find "Harley Hahn" you can search for "harley hahn", or "Harley Hahn", or "HARLEY HAHN".

White Pages Directories

A *white pages directory* is a service that allows you to search a database for someone's name or electronic mail address. White pages directories often contain other information as well, such as a postal address, a department name, a telephone number, and so on. For the most part, using a white pages directory is simple: you enter a name and, if it is in the directory, you will be shown whatever information is known about that name.

The name "white pages directory" was coined to suggest the image of a computerized telephone book. However, the analogy is misleading. All telephone books work pretty much the same way and contain the same standard information. On the Internet, there are a variety of white pages directories and

they do not all work the same. In this section, we will discuss the most well-known white pages directories and how you can use them.

The Gopher

If you know someone's organization, especially if he or she is at a university, the best place to start is the Gopher. Many organizations have CSO name servers or some other type of white pages directory. (We explain all about the Gopher and CSO name servers in Chapter 21.)

Thus, start by looking for a Gopher server for that organization. If you find one, look for a menu item that contains some type of directory, such as a CSO name server or an item that refers to a "phone book". For example, if you look at the main menu for the Gopher at the University of Wisconsin at Madison, you will see the item:

```
Phone books/
```

Selecting this menu leads you to a number of white pages directories.

Another alternative is to connect to the Gopher at Notre Dame University, in the state of Indiana. (The address is **gopher.nd.edu**.) From the main menu, select:

```
Non-Notre Dame Information Sources/
```

From the next menu, select:

```
Phone Books--Other Institutions/
```

You will now see a menu that offers several categories of white pages directories (which they call "directory servers"). From here you have access to hundreds of different white pages directories, not only for universities, but for all kinds of organizations.

Whois Servers

A *Whois server* is another type of white pages directory that allows you to query a database of names and electronic mail addresses, usually for a particular organization. There are a large number of Whois servers on the Internet.

You can get a list of Whois servers via Anonymous FTP. To find which Anonymous FTP hosts carry this list, use Archie to perform a **sub** search for **whois-servers**. (We discuss Anonymous FTP in Chapter 16 and Archie in Chapter 17.) The most well-known Whois server is **whois.internic.net**, which contains a lot of Internet names and addresses.

There are two main ways to access a Whois server. First, your system may have a command called **whois**. If so, you can type the name of the command, followed by **-h**, followed by the name of a particular Whois server, followed by the name of the person you are looking for. For example, to look for someone whose last name is Helliwell using the Whois server at **whois.internic.net**, enter:



```
whois -h whois.internic.net helliwell
```

(The **-h** stands for "host name".)

If you want to search for a first and last name, for example, Catherine Helliwell, use the following format:



```
whois -h whois.internic.net 'helliwell, catherine'
```

Be sure not to leave out the quotes. Notice that Whois servers do not distinguish between upper- and lowercase letters.

If you use the **whois** command without the **-h** and a host name, it will use a default Whois server. If your organization has its own Whois server, this may be the default. Otherwise, it will probably be a well-known server like **whois.internic.net**.

For example, if your organization has a Whois server, you should be able to search for all the people whose last name is Smith by using:



```
whois smith
```

If you are using a Unix system that has a **whois** command, you can read the documentation by using the **man** command to display the **whois** entry in the online Unix manual:



```
man whois
```

The second way to use a Whois server is to telnet to it. If the server provides public access, you should be able to log in with a userid of **whois**. For example, to use **whois.internic.net**, enter the command:

```
telnet whois.internic.net
```

Log in as **whois** (or follow whatever instructions you see). When you see the **Whois:** prompt, enter whatever name you want to search for. If you need help, enter **help**. (We explain how to use Telnet in Chapter 7.)

The Knowbot

A *knowbot* is an automated robot-like program that intelligently searches for information on your behalf. For instance, if you had a knowbot, you could tell it that you were looking for a recipe for vegetarian meatloaf with no radishes, and it would search the appropriate places on the Internet for you.

People have been talking about knowbots for some time now, but, as of yet, they are a dream unrealized. However, there is a primitive knowbot-like program that you can use to search many white pages directories at once, including one that contains people who use MCI Mail.

To use this knowbot, telnet to **nri.reston.va.us** using port number **185**:

```
telnet nri.reston.va.us 185
```

Once you are connected, enter **?** to display a summary of commands. It may take you a little while to learn how to use the knowbot effectively, but it is worth knowing. The basic command to use is **query**. For example, to search for someone whose name is Catherine Helliwell, enter:

```
query catherine helliwell
```

If you have an idea where the person may be, you can use some of the other knowbot commands to narrow down the search.

Fred and the X.500 Directories

X.500 is an international white pages directory that was developed to provide a universally accepted system for keeping track of people. Well, we are still waiting for X.500 to be universally accepted. In the meantime, a number of organizations have adopted the scheme, and you can search some of their directories by using a program called Fred.

To do so, telnet to either **wp.psi.com** or **wp2.psi.com** and log in as **fred**. (You will not need a password.) For example:

```
telnet wp.psi.com
```

Once you log in, enter **help** for a list of commands. For more detailed documentation, enter **manual**.

By the way, the name "Fred" stands for "Front End to the Directory".

The Usenet Address Server

You may already know the computer **rtfm.mit.edu** as the main repository of Usenet archives. For example, it contains copies of all the frequently asked question (FAQ) lists (see Chapter 9).

This computer is administered by Jonathan Kamens, who has set up an important white pages directory called the *Usenet address server*. A program on **rtfm.mit.edu** routinely scans each article that is submitted to Usenet looking for the **From** header line. (As we explained in Chapter 10, the **From** header line shows the name and address of the person who sent the article.) The program extracts all these **From** lines, and saves them in a database. Thus, if the person you are looking for has posted a Usenet article within the last year, there is a good chance that **rtfm.mit.edu** will have his or her name and mail address.

To use this service, mail a message to:

```
mail-server@rtfm.mit.edu
```

Leave the **Subject** line blank. In the body of the letter, type a line with the following format:

```
send usenet-addresses/name
```

where *name* describes the person you are looking for. The program on **rtfm.mit.edu** will search the Usenet address database and send back all the lines that contain the pattern you specified. Thus, you can use a first name, last name, or userid. Moreover, you do not need to use capital letters as the search program does not distinguish between upper- and lowercase.

As an example, we decided to search for Michael Peirce in Ireland by sending the following message:

```
To: mail-server@rtfm.mit.edu
Subject:
send usenet-addresses/peirce
```

In a short time, we received mail that contained 21 different lines containing **peirce** or **Peirce**, two of which referred to a Michael Peirce. By searching for the line with an Irish address (one that ends with the domain **ie**), we were able to find the person we wanted.

Hint

You can send more than one request per message. If you are not sure how to spell someone's name, send more than one request with alternate spellings. For example:

```
mail mail-server@rtfm.mit.edu
Subject:
send usenet-addresses/peirce
send usenet-addresses/pierce
send usenet-addresses/pirce
```

Each request will be answered in a separate message.

Another way to use the Usenet Address Server is via Wais. This allows you to search the database directly. All you need to do is start Wais and select the database called **usenet-addresses**. The nice thing about using Wais is that your searching can be more complex. For example, you can specify more than one word to search for and Wais will rank the responses in order of relevancy. (We explain how to use Wais in Chapter 23.)

Netfind: The Program

Netfind is a program that will actively search through the Internet, looking for a computer that knows about the person you are trying to find. To use Netfind, you need to have some idea of where the person is.

Netfind will not only look for a name and mail address, it will try to find Finger information about the person (see Chapter 8). If the person seems to have

more than one mail address, Netfind will make an educated recommendation as to which is the best one to use.

There are two ways to use Netfind. First, you may have a Netfind program on your computer. If not, you can telnet to a public Netfind host and use the program there. We will explain how to do this in the next section. (If this is the case for you, you should still read this section to see how Netfind works.)

If you have a Netfind program on your computer, you will be able to display the documentation from the online Unix manual—Netfind only runs on Unix systems—by using the command:

```
man netfind
```

You will see that the **netfind** program is complex and has all kinds of options. We will describe how to use it in its simplest sense, which will work fine most of the time.

Enter the **netfind** command, followed by a single name, followed by one or more locations. The name can be a first name, last name, or userid. The locations can be geographical or domains from an Internet address. Netfind does not distinguish between upper- and lowercase letters.

For example, let's say that you are looking for John Navarra, at Northwestern University in Evanston, Illinois. Here are several commands that you could try:

```
netfind navarra evanston
netfind navarra evanston illinois
netfind navarra northwestern
netfind navarra northwestern university
```

You could, of course, look for **john**, but there are probably a lot of Johns, and you are only allowed to specify one name.

Netfind works especially well if you know all or part of the person's address. For example, say that you happen to know that John Navarra has a computer account with Academic Computing and Network Services at the university. You also know that such accounts have mail addresses that end with **acns.nwu.edu**. You can use the following command:

```
netfind navarra acns nwu edu
```

Notice that you type each part of the address separately, leaving out the . characters.

As Netfind works, it will display a lot of technical messages showing its progress. Most of the time, these messages just get in the way. You can turn them off by using the option **-dfmn** just after the command name. For example:

```
netfind -dfmn navarra northwestern
netfind -dfmn navarra acns nwu edu
```

(The meaning of this option is described in the documentation. Essentially, what it does is turn off certain debugging information.)

When Netfind starts a search, it looks at the part of the Internet that you described and checks how many computers there are. If there are more than a few, Netfind will tell you that there are too many "domains". It will then list the computers that look promising and ask you to choose which ones you would like to search. For example, we enter the command:

```
netfind -dfmn navarra northwestern
```

Here are the first few lines of the output:

```
There are too many domains in the list.
Please select at most 3 of the following:

0. acns.nwu.edu (academic computing and network services,
   northwestern university, evanston, illinois)
1. astro.nwu.edu (astronomy department, northwestern university,
   evanston, illinois)
2. biochem.nwu.edu (biochemistry department, northwestern
   university, evanston, illinois)
3. cas.nwu.edu (northwestern university, evanston, illinois)
```

In this case, we know enough to choose number 0. Otherwise, we would have to guess.

Hint

There are two problems that you will encounter with Netfind's output. First, Netfind does not break up long lines as nicely as we did in our example, so the output usually looks ugly. Unfortunately, there is nothing you can do about it.

Second, Netfind will often generate so much output that it scrolls off your screen before you can read it. The best thing to do is use whatever facilities exist on your system to handle fast output. For example, on a Unix system, you can press CTRL-S to pause the output and CTRL-Q to continue. On a PC, you can press the PAUSE key to pause and any other key to continue. If you are using Netfind within a window under X Window, you can use the scroll bar to look back at the previous output.

Once you tell Netfind what it needs to know, it will proceed with an intelligent search of all the computers in the appropriate part of the Internet. In our example, Netfind will search all the computers that fall under the domain of Academic Computing and Network Services at Northwestern University. (That is, all the computers whose addresses end with **acns.nwu.edu**.) Netfind will attempt to connect to each computer looking for information about a user whose name or userid is **navarra**.

When it finds such a person, it will request the Finger information and display it for you. You will frequently find that a person has Finger information on more than one computer—say, on every computer in a departmental network. In such cases, Netfind will display a plethora of information, and you will have to figure out which parts are the most valuable.

If possible, Netfind will look at the times that the person has last logged in to each computer and deduce which of several mail addresses would be the best to use. (For an explanation about domains and addresses, see Chapter 4. For information about interpreting Finger information, see Chapter 8.)

Hint

You can interrupt a Netfind search by pressing CTRL-C.

Netfind: The Server

If your computer does not have a Netfind program, you can telnet to one of the public Netfind servers listed in Table 22-1 and log in as **netfind**. You will not need a password.

When you use the Netfind server, you make selections from a menu to tell Netfind what to do. When you get to the point where you must tell Netfind what to search for, enter the same type of information that we discussed in the previous section.

After logging in as **netfind**, you will see the following main menu:

```
Top level choices:
        1. Help
        2. Search
        3. Seed database lookup
        4. Options
        5. Quit (exit server)
```

Select number 2. You will now see:

```
Enter person and keys (blank to exit) -->
```

You can now tell Netfind what to search for. For example, you might enter:

```
navarra acns nwu edu
```

Location	Internet Address	IP Address
Australia	archie.au	139.130.4.6
Canada	macs.ee.mcgill.ca	132.206.61.15
Chile	malloco.ing.puc.cl	146.155.1.43
Czech Republic	netfind.vslib.cz	147.230.16.1
England	monolith.cc.ic.ac.uk	155.198.5.3
Singapore	lincoln.technet.sg	192.169.33.6
Slovakia	nic.uakom.sk	192.108.131.12
South Korea	nic.nm.kr	143.248.1.100
USA: Alabama	redmont.cis.uab.edu	138.26.64.4
USA: Colorado	bruno.cs.colorado.edu	128.138.243.151
USA: Minnesota	mudhoney.micro.umn.edu	134.84.132.7
USA: Texas	netfind.oc.com	192.82.215.92
USA: Virginia	ds.internic.net	198.49.45.10
Venezuela	dino.conicit.ve	150.188.1.10

Log in with a userid of **netfind**.

Table 22-1. *Public* **netfind** *Servers*

Netfind will perform its search and display its output in the same manner that we have already discussed.

> **Hint**
>
> You can interrupt a Netfind search by pressing CTRL-C.

When your search is finished, you will once again see the message:

```
Enter person and keys (blank to exit) -->
```

Press RETURN and you will be back at the main menu. You can now select number 5 to quit.

Using Netfind to Find the Location of a Computer

There will be times when you know the address of a computer and you want to find out where it is. Or you may want to find all the computers in a certain area. You can use a Netfind server for both of these tasks.

Here is how it works. Telnet to one of the public Netfind servers in Table 22-1 and log in as **netfind**. You will see the main menu:

```
Top level choices:
        1. Help
        2. Search
        3. Seed database lookup
        4. Options
        5. Quit (exit server)
```

Select number 3 (the *seed database* is the list of computers and locations that Netfind uses for its searches). You will see another menu:

```
Seed database choices:
        1. Seed database help
        2. Seed database search
        3. Toggle seed database search output format
        4. Quit menu (back to top level)
```

Select number 2. You will see the following:

```
Keys (blank to exit):
```

You can now enter the address of a computer or a geographical location. If you enter the address, use only the rightmost two domains and leave out the . character. Remember, Netfind does not distinguish between upper- and lowercase letters.

For example, say that you want to find out the location of the computer **rtfm.mit.edu**. Enter:

```
mit edu
```

In this case, you would see a list of computers in that part of the Internet. In particular, you will see:

```
mit.edu massachusetts institute of technology, cambridge
```

This is enough to show you that **rtfm.mit.edu** is probably in Cambridge, Massachusetts.

Here is another example. Say that you want to look for computers in Dublin, Ireland. Tell Netfind to search for:

```
dublin ireland
```

You will see a long list of computers. Be aware that these are not all the computers in Dublin, but only the ones Netfind knows about.

When you are finished, press RETURN to get back to the menu, and then work your way back out by choosing "Quit". (It will be obvious.)

What to Do When All Else Fails

If you have tried everything you can think of to find someone and you still can't get anywhere, there are several alternatives.

If you know a person's userid and address—or if you can make a good guess—you can use the Finger service to display whatever public information is offered about that person. (See Chapter 8.)

Second, you can post an article to the Usenet newsgroup named **soc.net-people**. One of the purposes of this group is for people to ask questions like, "Does anyone know where to find Chuck Wagon, who graduated from the University of Foobar in 1983 from the Unix Studies department?" When you send the article, make sure that the **Subject** line has the person's name and location (if you know it).

If you are looking for someone at a university, there is an article by David Lamb of Queens University in Canada that you will find helpful. The article has a summary of various methods of finding people, followed by specific information about many different universities.

Lamb's article is posted regularly to the **soc.net-people** newsgroup in three parts. You can also obtain it via Anonymous FTP from **rtfm.mit.edu**. Change to the directory named **/pub/usenet/news.answers/mail/college-email** and download the three files named **part1**, **part2**, and **part3**. The command **mget part*** will do the job. (We explain how to use Anonymous FTP in Chapter 16.)

Another article to read is called "Finding Addresses". It was written by Jonathan Kamens (the keeper of the Usenet Archive at **rtfm.mit.edu**). This article describes many different ways to find someone, including some esoteric techniques that we did not cover in this chapter. The article is also posted regularly to **soc.net-people**. Or, you can get it via Anonymous FTP from **rtfm.mit.edu**. Change to the directory named **/pub/usenet/news.answers** and download the file named **finding-addresses**. (In passing, we would like to mention that the same directory contains an article called **finding-sources** that will show you how to find computer programs in source code form.)

If you know the name of the computer on which someone has an account, you can send mail to the userid named **postmaster** at that address and ask for help. For example, if you were looking for someone who uses the computer named **fuzzball.ucsb.edu**, you could send mail to:

```
postmaster@fuzzball.ucsb.edu.
```

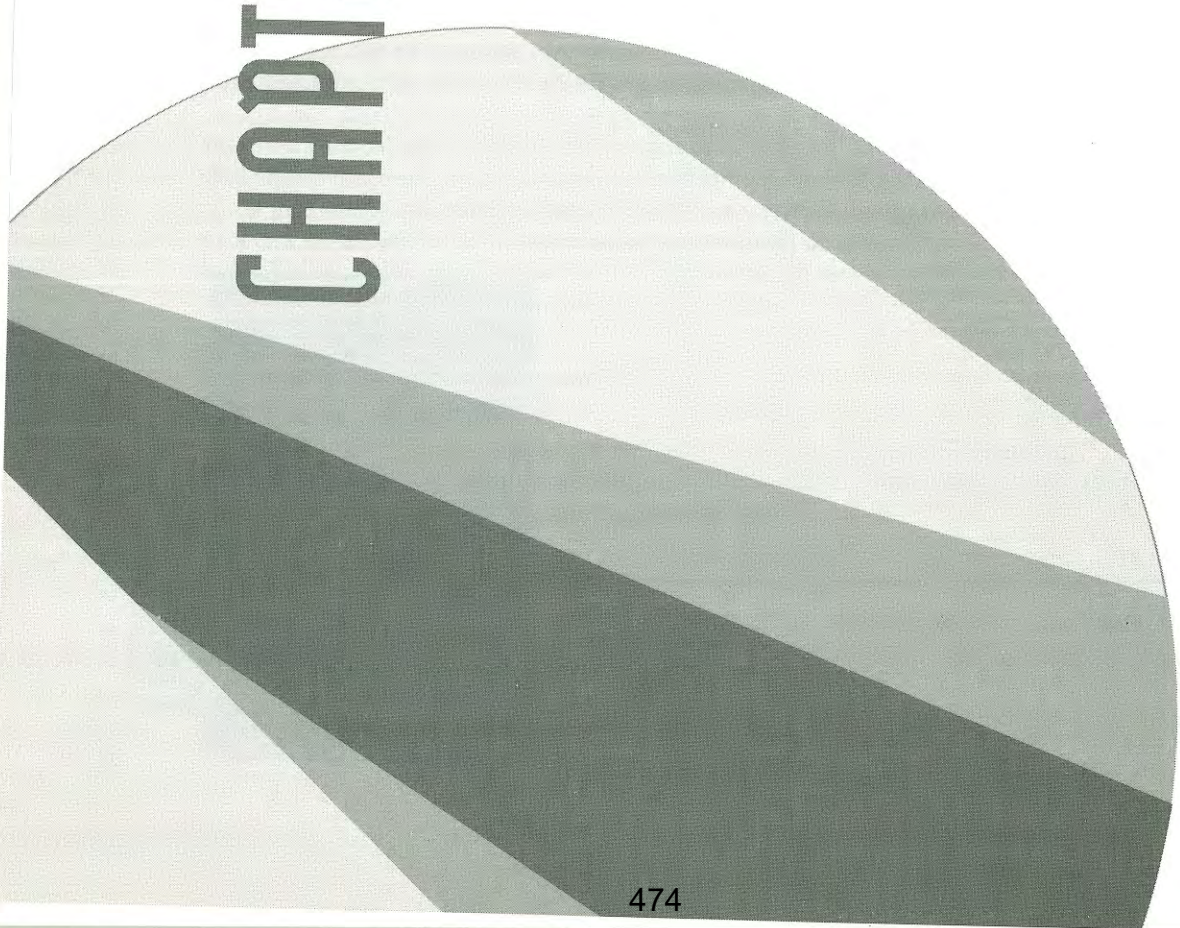
Most Internet hosts have someone who answers such mail. Remember, though, this will require a busy person to take the time to answer you, so make sure you have tried the other alternatives first.

Hint

When nothing else has worked, try the telephone.



CHAPTER 23



Wais

Wais is a tool that allows you to search large amounts of information quickly and thoroughly.

Of course, such tools are only as good as the quality and variety of the information, as well as the power of the searching program. As you will see, Wais can search any of several hundred information sources on a large variety of topics.

Nevertheless, Wais is far from perfect, and we will discuss its limitations. But, within these limitations, Wais is an amazing tool. Once you learn how to use it, Wais is like having a team of simple-minded, but efficient librarians, ready to search for whatever it is you need, whenever you need it.

475

The Motivation for Wais

The name Wais (pronounced "Wayz") stands for "Wide Area Information Service". As an Internet service, Wais is unique in that it grew out of a project started by three commercial companies: Apple, Thinking Machines, and Dow Jones.

The original idea behind Wais was an ambitious one. In a world of too much information, a computer could keep track of a vast amount of data, sift it for you, and present you with only the information that is relevant to your needs. The idea was to create a program that would act as a personal reference librarian, saving you time and making it possible to access information that you might never even know existed.

For example, say that your daily newspaper was delivered to you in electronic form, as information that you could display on your computer screen. Wouldn't it be nice if you could tell a program what you wanted to see and have it do all the work?

For example: "Each day, I want you to show me sports news, but not business. I want to see anything that has to do with baseball, but only if it mentions the New York Mets. And I don't want to see any football stuff, except when it's time for the Super Bowl. Oh yes, I also want to see the world news, but not too much politics, but if there is anything about airplanes, be sure to include it..."

You get the idea: let a computer program do the work. Moreover, when the program finds an article that really interests you, you should be able to tell it to keep an eye out for other articles on the same subject.

Of course, once you have such a system, you do not have to confine yourself to news. Any type of data that can be accessed by computer can, in principle, be searched on your behalf by a discriminating computer program.

You can see the motivation here for Wais: Apple sells personal computers with an easy-to-use graphical interface; Thinking Machines makes computers that contain many processors, suitable for searching large amounts of data quickly; and Dow Jones runs a service that sells news and information.

Obviously there are some wonderful aspects to such a system. You can let a computer program handle all of the details of searching for information. All you have to do is sit back like an aristocrat, reading your daily personalized newspaper.

Still, there are important limitations. First, a series of articles on your computer screen, no matter how personalized, will never take the place of an actual newspaper or book that you can lay flat on the table and scan with your eye, page by page.

Second, whatever data you receive is only as good as the questions that you ask, plus whatever the computer program can infer about your likes and dislikes. All too often, it is impossible to ask exactly the right questions, even though you may know what you want when you see it.

Finally, the computer program can only find data that is computerized, which limits things enormously. For example, think about how much information you can access in a regular library. You can walk into a library, take any book off the shelf, open the book and read it. Although you may sometimes have trouble finding exactly what you want, you do have access to every book on the shelves, as well as a catalog and, possibly, a human reference librarian.

Hint

One mistake that people often make is thinking that just because they can search for a book in a computerized library catalog, they should be able to read the text of that book using the computer. In most cases, even if you use a computer to help you find a book, you will still have to get the real thing in order to read it.

Compared to a real library, there are very, very few books and periodicals whose text is stored on a computer.

The original idea behind Wais was to develop a generalized system of information retrieval that would be able to access collections of data all around the world. Some of the data would be available for free over the Internet, but it wouldn't be long before people would start to pay for information.

What happened to Wais is somewhat predictable, at least in hindsight. It did not turn out to be the information utility of the future. Yes, there are some people who do use Wais-like computer systems for which they pay real money, but most of us still depend on old-fashioned paper.

So, with respect to the Internet, Thinking Machines has stopped supporting the publicly distributed Wais. A free version of Wais, called *Freewais*, is now maintained by an organization called the CNIDR (Clearinghouse for Networked Information Discovery and Retrieval).

Nevertheless, Wais is important to the Internet. Why? Because it evolved into a system that can perform one vital service extremely well.

What Is Wais?

Wais is an Internet service that can search any of hundreds of collections of data. Each such collection is called a *source*. You tell Wais which sources to use and

what you would like to find. To do so, you specify one or more words, called *keywords*, for which you want Wais to search. Wais will search the entire text of each item in the sources you specify, and find the ones that meet your criteria. This is called a *full-text search.*

For example, you can point Wais at a large set of recipes and tell it to find all the recipes that contain the words **garlic**, **chicken**, and **rice**. Or you can search a collection of Bill Clinton's speeches from the 1992 U.S. presidential campaign and find all the speeches that mention **tax**, **cut**, and **promise**. Or you can search a reference of protein sites and patterns, and select all the citations that contain the words **dihydrofolate reductase**. Or you can search the Bible, the Quran, and the Book of Mormon, looking for all the passages that contain the word **adultery**. (Think how much time this could save you.)

Using Wais is reasonably simple. All you do is select one or more sources and then tell Wais what to search for. Wais will connect to the computers that contain the sources and ask them to carry out the search. The output will be a list of articles or citations.

Wais will display the list for you, showing the items from most to least relevant. What Wais thinks is relevant is based on how often the keywords you specified showed up in each article. Wais considers the most relevant articles to be the ones with the most occurrences of the keywords.

Once Wais shows you what it has found, you go through the list and pick out which items you want to look at. Wais will then retrieve the actual text and display it on your screen. If you would like to keep the item, you can tell Wais to save it to a file or mail it to yourself (or to someone else).

When you find an article that is especially germane to your interests, you can tell Wais to scan all or part of the article and to use those words for future searches. This is called *relevance feedback.*

The service that Wais provides can be extremely useful. Still there are limitations. First, Wais can only help you if it has access to the information you want. Although several hundred information sources sounds like a lot, they do not begin to cover all the questions a human being might have.

Second, Wais searches virtually every word contained in a source—ignoring small, common words like "a" and "the"—but only in a simple, non-contextual manner. For instance, you can tell Wais to search a collection of speeches for the words **tax**, **cut**, and **promise**, but those words might be anywhere in the text. For example, the person giving the speech might have said, "We must tax our ingenuity to cut down on our inability to fulfill the promise of the future." There is no way to tell Wais to find references to "promises about tax cuts".

Still, when it works well, Wais is a fabulous tool that can save you many hours of research. So let's take a look at how it works and how to use it.

Wais Clients

Like many Internet services, Wais uses a client/server system (see Chapter 2). You interact directly with a *Wais client* program. It is this program that displays information and carries out your commands. Each Wais information source is maintained by a *Wais server* program. There are many public Wais servers, all over the Internet, and all you need to be able to access them is a Wais client.

Whenever you make a request, your Wais client connects to the appropriate Wais server and asks it to perform a search. The server carries out the search, and returns the results to the client, which displays the results on your screen. When you indicate that you would like to view a particular item, your client again sends the request to the server. The server sends back the item, which your client displays, one screenful at a time.

There are a variety of Wais clients available for free, via Anonymous FTP. It may be that your system already has a Wais client. If so, all you have to do is start the program. Two of the most common Wais clients are **swais** and **waissearch** for regular Unix systems, and **xwais** for X Window systems. (We discuss X Window in Chapter 2.) There are also Wais clients for a large number of other systems.

If you do not have a Wais client on your computer, you can telnet to one of the hosts listed in Table 23-1 and log in as indicated. Each of these computers provides a public Wais client that anyone can use. Once you log in, the client will start automatically. No password is necessary. (We discuss Telnet in Chapter 7.)

Hint

The public Wais hosts do not all offer the same set of sources. You may want to try the various hosts to see what is available.

When you telnet to a public Wais client, you will be using the **swais** client. (The name means "simple Wais".) For the rest of the chapter, we will assume that this is the client you are using. If you are using a different client, you will have to read the appropriate documentation. However, the basic concepts as to how to use Wais will be the same as what we discuss.

For more information about Wais, there are two Usenet discussion groups you can subscribe to: **comp.infosystems.wais** and **alt.wais**. There is also a Wais frequently asked question (FAQ) list that you will find helpful, especially if you are looking for a Wais client for your computer.

Location	Internet Address	IP Address	Log in as...
Finland	info.funet.fi	128.214.6.102	wais
USA: California	swais.cwis.uci.edu	128.200.15.2	swais
USA: Massachusetts	nnsc.nsf.net	128.89.1.178	wais
USA: Massachusetts	quake.think.com	192.31.181.1	wais
USA: North Carolina	kudzu.cnidr.org	128.109.130.57	wais
USA: North Carolina	sunsite.unc.edu	152.2.22.81	swais

Table 23-1. *Public Wais Clients to Which You Can Telnet*

The FAQ list is posted regularly to **comp.infosystems.wais**. It can also be downloaded via Anonymous FTP from **rtfm.mit.edu**. Look in the directory named **/pub/usenet/news.answers/wais-faq** for a file called **getting-started**. (We discuss Usenet newsgroups and FAQ lists in Chapter 9. We discuss Anonymous FTP in Chapter 16.)

Musing on Wais

In the following sections, we will show you how to use a public access Wais client to perform a search. Before we do, we have a few words of advice.

It can take a while to get good at using Wais. Your results depend very much on how good you are at choosing which sources to search, and how good you are at formulating keywords. Moreover, the Wais interface is kind of funky and takes some getting used to. (For those of you who do not live in Southern California, funky means "low quality, but still cool".)

There are nuances about Wais that you must come to understand over a period of time, akin to those that a good librarian develops when it comes to knowing where to look for just the right book. Don't be frustrated if, at the beginning, you find it difficult to come up with anything useful. Once you get some experience, it becomes a lot easier to use Wais effectively.



FUN TIP: From time to time, you may find yourself in the position of having to put on a short demonstration to convince a skeptical person that all the time you spend on the Internet is worthwhile. For example, you may be called upon to demonstrate the Internet for the big boss, or for the guy in the next office who you have been trying to impress for the last six months, or for a well-meaning friend who thinks computers are generally a waste of time.

Our advice is to show them the Gopher, which is a lot more reliable and easier to use than Wais. If you decide to show off by using Wais, that will be the day that the client program is slow and uncooperative, and that every source you search will be temporarily not working. Instead of showing them how well Wais can find data, you will probably end up with nothing but an embarrassed look on your face.

Okay, having offered the preceding caveat, let's see what it looks like to use Wais.

Starting to Use Wais

To start Wais, you can telnet to one of the public clients. For example:

```
telnet quake.think.com
```

As the Telnet connection is made, you will see the following messages. (For an explanation of how to use Telnet, see Chapter 7.)

```
Trying 192.31.181.1...
Connected to quake.think.com.
Escape character is '^]'.

SunOS UNIX (quake)
login:
```

Log in as **wais** (you will not need a password). You will then see:

```
Last login: Sat Jul 10 15:32:36 from jazz.ucc.uno.edu
SunOS Release 4.1.1 (QUAKE) #3: Tue Jul 7 11:09:01 PDT 1992

Welcome to swais.
Please type user identifier (optional, i.e user@host):
```

At this point, you should enter your Internet mail address and press RETURN. The remote host now asks what type of terminal you are using:

```
TERM = (vt100)
```

You can either enter the type of terminal or press RETURN to indicate a VT-100. (We discuss terminals in Chapter 3.) Once you answer the question, you will have to wait a moment for the **swais** client to start. When it starts, you will see a list of sources. You are now ready to start work.

```
SWAIS                    Source Selection              Sources: 463
  #          Server                    Source                  Cost
001:  [          archie.au] aarnet-resource-guide                Free
002:  [    munin.ub2.lu.se] academic_email_conf                  Free
003:  [wraith.cs.uow.edu.au] acronyms                            Free
004:  [   archive.orst.edu] aeronautics                          Free
005:  [ ftp.cs.colorado.edu] aftp-cs-colorado-edu                Free
006:  [nostromo.oes.orst.ed] agricultural-market-news            Free
007:  [   archive.orst.edu] alt.drugs                            Free
008:  [    wais.oit.unc.edu] alt.gopher                          Free
009:  [sun-wais.oit.unc.edu] alt.sys.sun                         Free
010:  [    wais.oit.unc.edu] alt.wais                            Free
011:  [alfred.ccs.carleton.] amiga-slip                          Free
012:  [    munin.ub2.lu.se] amiga_fish_contents                  Free
013:  [  coombs.anu.edu.au] ANU-Aboriginal-Studies          $0.00/minute
014:  [  coombs.anu.edu.au] ANU-Asian-Computing             $0.00/minute
015:  [  coombs.anu.edu.au] ANU-Asian-Religions             $0.00/minute
016:  [       150.203.76.2] ANU-CAUT-Academics              $0.00/minute
017:  [  coombs.anu.edu.au] ANU-CAUT-Projects               $0.00/minute
018:  [  coombs.anu.edu.au] ANU-Coombspapers-Index          $0.00/minute

Keywords:

<space> selects, w for keywords, arrows move, <return> searches, q quits, or ?
```

Let's take a look at the various parts of the screen. First, the top line tells us that this is the Wais source selection screen. (Remember, we have to select our sources before we can search.) On the top right, we see that this client knows about 463 different sources.

Following this, we see a summary of the first 18 sources. Each line shows the address of the host that contains the source, the name of the source itself, and the price of performing a search. Notice that the first source is highlighted. As you use this list, you will move from one source to another. The highlighting will move also, to indicate the *current source*.

Hint

Don't worry about the price. All the sources are free. You cannot spend money by accident.

On the second to last line we see **Keywords:**. This is where we will type our keywords when we are ready to begin searching.

Finally, the last line summarizes the basic **swais** commands. These will make more sense once you learn how to use **swais**.

In the next few sections, we will show you how to select sources and how to perform a search. Before we do, we would like to introduce you to the two most basic **swais** commands.

First, you can display a summary of commands by pressing **h** (help) or **?**. The **swais** client works in what is called *cbreak mode*. This means that you do not have to press RETURN when you use a single character command. For example, when you want help, you only need press the **h** key. Do not press RETURN.

Second, to stop **swais** and disconnect from the remote host, press **q** (quit).

Hint

Be careful with the **q** key. It is all too easy to press **q** inadvertently—say when you are reading the list of sources—and irrevocably abort your session with Wais.

Selecting a Source

The general plan for using Wais is to select one or more sources, and then perform a search. In this section, we will go over the commands that you use to select a source. In the next section, we will show you how to perform a search. We will then present an example that illustrates how it all works.

To select a source, you move to it and then press SPACE. Wais will show you that a source is selected by marking it with a * character. For example, in the following list, source number 10 is selected:

```
008:   [    wais.oit.unc.edu] alt.gopher                Free
009:   [sun-wais.oit.unc.edu] alt.sys.sun               Free
010: * [    wais.oit.unc.edu] alt.wais                  Free
011:   [alfred.ccs.carleton.] amiga-slip                Free
```

You can select as many sources as you want before you start to search. At any time, you can select a new source by moving to it and pressing SPACE. If you decide to stop using a source that is already selected, move to it and press SPACE once again. This will unselect the source. To unselect all the sources and start fresh, press = (equal sign).

Hint

Before you start a new search, press = to clear any sources that may have been selected from the previous search.

There are several ways to move from one line to another. The easiest way is to use the UP and DOWN keys (the cursor control keys, with the arrows). If these do not work on your terminal, or if you prefer not to use them, you can use **j** or CTRL-N to move down to the next line, and **k** or CTRL-P to move up to the previous line. (Although this choice of keys may seem odd, they are similar to what is used with the **vi** editor and are second nature to many Unix users.)

To move down to the next screenful of sources, you can use either **J** or CTRL-D. To move up to the previous screenful of sources, you can use either **K** or CTRL-U. (Note: these are the uppercase "J" and "K" letters.)

To jump directly to a particular source, simply enter its number. For example, to jump to source number 15, enter:

```
15
```

(You do not have to type the leading **0** in **015**.)

If you know the name of a source, you can jump to it by entering a / (slash) character, followed by the first few letters of its name. For example, to jump to the source named **amiga_fish_contents**, it is enough to enter:

```
/amiga_f
```

If you would like to see the technical information that Wais knows about a source, move to it and press **v** (view). For example, if you move to **amiga_fish_contents** and press **v**, you will see the following:

```
Name:           amiga_fish_contents.src
Directory:      /sources/
Maintainer:     hakan@hera.dit.lth.se
Selected:       No
Cost:           Free
Server:         munin.ub2.lu.se
Service:        210
Database:       amiga_fish_contents
Description:
Server created with WAIS release 8 b5 on Aug 29 16:24:34 1992
   by anders@munin

This is an index of the contents of Fred Fish's disks #1-current
with a freely distributable AMIGA software library containing an
extensive collection of PD, shareware and demo programs.

Search for 'disknr' to see the current (latest) disk number.
Search for 'ftp' to see some ftp sites where you can get the
  disks.
```

Performing a Search

Once you have selected one or more sources, you are ready to perform a search. To do so, press RETURN. The cursor will move to the line near the bottom of the screen.

```
Keywords:
```

You can now type whatever keywords you want. As you type, there are several ways for you to make corrections.

To erase a single character, press BACKSPACE. If this does not work, you can also try CTRL-H, DELETE, or CTRL-BACKSPACE. For reasons we won't go into, one of these keys will work.

To erase an entire word, press CTRL-W. To erase the entire line, press CTRL-U.

Here is an example of what the line looks like after we have entered several keywords:

```
Keywords: tax cut promise
```

Once you have specified your keywords, press RETURN to begin the search. Wais will check each source that you specified and display a list of possibilities. Wais will organize this list so that the best matches come first. Here is an

example that we generated by searching the source named **clinton-speeches** for the above-mentioned keywords:

```
SWAIS                                  Search Results                    Items: 40
  #    Score     Source                        Title                          Lines
001:   [1000] (clinton-speeche)     VP DEBATE ANALYSIS: Encyclopedi          697
002:   [ 928] (clinton-speeche)     THE ECONOMY: Statement                   791
003:   [ 928] (clinton-speeche)     THE ECONOMY: Speech - Detroit,           559
004:   [ 819] (clinton-speeche)     LABOR: Speech - Washington, DC           525
005:   [ 782] (clinton-speeche)     ECONOMIC STRATEGY: 6/21/92               877
006:   [ 728] (clinton-speeche)     ECONOMIC PLAN: Position Paper -          773
007:   [ 691] (clinton-speeche)     EDUCATION: Speech - Rockville,           658
008:   [ 673] (clinton-speeche)     VARIOUS TOPICS: Interview - Atl          830
009:   [ 655] (clinton-speeche)     ECONOMICS: "New Covenant" Speec          634
010:   [ 637] (clinton-speeche)     SMALL BUSINESS PLAN: Position P          598
011:   [ 601] (clinton-speeche)     MANUFACTURING: Speech - Washing          706
012:   [ 600] (clinton-speeche)     ON RESPONSIBILITY (BUSH'S): Spe          237
013:   [ 600] (clinton-speeche)     THE ECONOMY: Speech - New Orlea          227
014:   [ 564] (clinton-speeche)     HEALTH CARE: Speech - Macon, GA          440
015:   [ 546] (clinton-speeche)     LABOR: Speech - San Diego, CA            507
016:   [ 509] (clinton-speeche)     ECONOMICS: Speech - Flint, MI -          283
017:   [ 509] (clinton-speeche)     JOB RETRAINING: Speech - Dayton          359
018:   [ 509] (clinton-speeche)     EDUCATION: Speech - Los Angeles          638

<space> selects, arrows move, w for keywords, s for sources, ? for help
```

Now that you have seen what Wais has found, you can decide if you want to look at anything. Notice the score in the second column. This gives you an indication of how well each item matched your keywords. The item that had the most matches is always given a score of 1000. All the other items are graded relative to the best match.

Hint

A score of 1000 does not mean that this is the best item to meet your needs. It simply means that—according to Wais' criteria—this particular item had the most keyword matches. You will still have to examine the results. In many cases, you will want to specify better keywords and try again.

At this point, you have two choices. You can look at some of the items, or you can try another search. To look at an item, move to it and press RETURN. Wais will connect to the appropriate server and ask for a copy of the actual file. When it arrives, Wais will use a paging program to display the file, one screenful at a time.

As you read a file, your screen is controlled by the paging program. At this point, there are two basic commands that you can use. To display the next screenful of text, press SPACE. To stop displaying the article, press **q** (quit).

Hint

While you are reading an article, pressing **q** will quit the paging program. At all other times, pressing **q** will quit Wais. You will have to get used to this.

If you would like to have a permanent copy of what you have just read, you can use the **S** (uppercase "S") command to save the text to a file, or the **m** command to mail the text. Of course, you can only use the **S** command if you are using a Wais client on your own computer. (You cannot save files on a public Wais host.) However, you can always use the **m** command to mail a copy of the text to yourself, or to someone else.

If you are not satisfied with the results of the search, you can try again by pressing **w** (keyword). This will bring back the keyword entry line. You can now change your keywords and start a new search.

When you decide that you are finished searching this particular set of sources, press **s**. This will bring back the sources screen. You can now select or unselect whatever sources you want. Remember, if you decide to start fresh with new sources, you can press = to unselect everything.

Until you perform a new search, you can re-display the results of the previous search by pressing **r**.

At this point, we would like to say a few words about relevance feedback. With some Wais clients, you can select all or part of an item as being representative of the type of thing you are looking for. You can then tell Wais to save all the words in the item to help with future searches. This is called relevance feedback.

When you telnet to a public Wais client, relevance feedback is not much of a consideration. However, if you are using a sophisticated Wais client on your own computer, you can use relevance feedback to help you create personalized searches of great specificity.

Hint

If you get serious about Wais, get yourself a good Wais client and spend some time with it.



FUN TIP: Wais is like a sewer. What you get out of it depends on what you put into it.

A Summary of Wais Commands

The summary in Figure 23-1 shows the basic commands for using Wais with the **swais** client program. If you are using another client, the commands will be different (and possibly more powerful), but the basic concepts will be the same.

A Real-Life Example of Using Wais

To complete this chapter, we will use Wais to search for some important information that will be of immediate practical benefit in your everyday life.

You have decided that you are tired of the pace of modern life and you want to go live on a small island, somewhere in the tropics. The only stipulation is that there must be plenty of coconuts and guano so that you can support yourself. The question is: where should you go? To find the answer, you use Wais.

To begin, you telnet to a public Wais client and log in. For this example, you use:

```
telnet nnsc.nsf.net
```

and log in as **wais**. Once you have specified your terminal type, the Wais client starts by displaying the list of sources.

To start, you need to know which source (or sources) to search. If you are intimately familiar with all the hundreds of different sources, this should not be much of a problem. However, if you are not sure which one is best, you can use Wais to help you find it.

There happens to be one particular source that contains information about all the other sources. It is called **directory-of-servers**. You can search this source to help you find other sources.

First, jump to this source by entering:

```
/directory-of-servers
```

Next, select this source by pressing SPACE. To tell Wais that you are ready to enter keywords, press RETURN. The cursor will now move to the keyword line.

Basic Commands

`h`	display summary of commands [help]
`?`	same as `h`
`q`	quit Wais

Moving the Current Line

DOWN	move down one line
`j`	same as DOWN
CTRL-N	same as DOWN
UP	move up one line
`k`	same as UP
CTRL-P	same as UP
`number`	jump to specified line number
/pattern	jump to source that begins with specified pattern
`J`	move down to next screenful
CTRL-D	same as `J`
`K`	move up to previous screenful
CTRL-U	same as `K`

Selecting a Source

SPACE	select or unselect a source
`=`	unselect all sources
RETURN	(after selecting sources) ask for new keyword
`r`	re-display results of the previous search
`v`	display technical information about source [view]

Performing a Search

RETURN	(after entering keywords) start search
RETURN	(while examining results of a search) display item
`w`	ask for new keywords
`s`	re-display the source screen

Reading an Article

SPACE	display next screenful
`q`	stop reading article [quit]

Figure 23-1. *Commands for using Wais with the* ***swais*** *client program*

Now, what keywords can you use to choose a source for finding some place on the Earth? Try the following:

```
world geography
```

Press RETURN to start the search. After a few moments, Wais displays the results of your search:

```
SWAIS                          Search Results                    Items: 24
         Score   _____Source_____   _____________Title_____________
--> 001  [1000] (directory-of-se)  world-factbook.src                  21
    002  [ 526] (directory-of-se)  eros-data-center.src                94
    003  [ 474] (directory-of-se)  CCINFO.src                         106
    004  [ 368] (directory-of-se)  unced-agenda.src                    71
    005  [ 368] (directory-of-se)  world91a.src                        26
    006  [ 316] (directory-of-se)  ANU-Pacific-Relations.src           90
    007  [ 316] (directory-of-se)  ANU-Thai-Yunnan.src                 83
    008  [ 316] (directory-of-se)  ASK-SISY-Software-Information.src   34
    009  [ 316] (directory-of-se)  Arabidopsis_thaliana_Genome.src     37
    010  [ 316] (directory-of-se)  Connection-Machine.src              25
    011  [ 316] (directory-of-se)  Func-Prog-Abstracts.src             23
    012  [ 316] (directory-of-se)  POETRY-index.src                    28
    013  [ 316] (directory-of-se)  Queer-Resources.src                 21
    014  [ 316] (directory-of-se)  US-State-Department-Travel-Advisorie 89
    015  [ 316] (directory-of-se)  comp.doc.techreports.src            18

<space> selects, arrows move, w for keywords, s for sources, ? for help
```

Wais has found a number of possible sources for you, but only number 1 even comes close to a score of 1000. To find out something about it, press RETURN. You see:

```
(:source
   :version  3
   :ip-address "131.239.2.100"
   :ip-name "cmns-moon.think.com"
   :tcp-port 210
   :database-name "CIA"
   :cost 0.00
   :cost-unit :free
   :maintainer "bug-public@think.com"
   :subjects "social sciences demographics politics CIA worldfact
book population economics imports exports business"
   :description "Connection Machine WAIS server.  The WorldFact
book by the CIA which contains a good description of every
country. The entry for WORLD is also particularly good.

Descriptions of 249 nations, dependent areas, and other entities
with information on population, economic condition, imports/exports,
conflicts and wars, and politics.  Produced annually by the CIA.
Search 'World Factbook' for table of contents.")
```

Well, isn't that nice of the CIA to provide you with such useful information for free? This sounds like just what you need. Press **q** (to end the paging program). You are now back at the results screen.

To prepare for your next search, you need to select a new source. Press **s** to return to the sources screen. Next, press = to unselect all selected sources. (In this case, we only had one source already selected, but pressing = before starting a new search is a good habit.)

Now we need to select the CIA World Factbook. To jump to this source, enter:

```
/world-factbook
```

To select it, press SPACE. To specify new keywords, press RETURN.

You will be back at the keyword line, but the previous keywords will still show. To erase them, press CTRL-U. Now, type the new keywords:

```
coconut guano
```

(Remember, we are looking for a tropical island with coconuts and guano.) To start the search, press RETURN. Here are the results:

```
SWAIS                          Search Results                    Items: 25
         Score   _____Source_____   _____________Title_____________
--> 001  [1000] (cmns-moon.think)  Glorioso Islands (French possession)    79
    002  [ 543] (cmns-moon.think)  Juan de Nova Island (French possessi    83
    003  [ 533] (cmns-moon.think)  Navassa Island (territory of the US)    80
    004  [ 515] (cmns-moon.think)  Johnston Atoll (territory of the US)   107
    005  [ 515] (cmns-moon.think)  Jarvis Island (territory of the US)     90
    006  [ 515] (cmns-moon.think)  Howland Island (territory of the US)    99
    007  [ 515] (cmns-moon.think)  Baker Island (territory of the US)      95
    008  [ 477] (cmns-moon.think)  Soviet Union  Geography Total area:    467
    009  [ 440] (cmns-moon.think)  Palmyra Atoll (territory of the US)     79
    010  [ 421] (cmns-moon.think)  Tonga  Geography Total area: 748 km2   286
    011  [ 421] (cmns-moon.think)  Niue (free association with New Zeal   271
    012  [ 412] (cmns-moon.think)  Western Samoa  Geography Total area:   288
    013  [ 412] (cmns-moon.think)  Saint Lucia  Geography Total area: 6   290
    014  [ 412] (cmns-moon.think)  Philippines  Geography Total area: 3   381
    015  [ 412] (cmns-moon.think)  French Polynesia (overseas territory   288

<space> selects, arrows move, w for keywords, s for sources, ? for help
```

Again, Wais has found a number of items, but only one has a high score. To display this item, press RETURN. Here is what you see (with some of the blank lines omitted):

```
CIA World Factbook       Glorioso Islands (French possession)

Geography
Total area: 5 km2; land area: 5 km2; includes Ile Glorieuse,
Ile du Lys, Verte Rocks, Wreck Rock, and South Rock

Comparative area: about 8.5 times the size of The Mall in
Washington, DC

Land boundaries: none
Coastline: 35.2 km
Maritime claims:
Contiguous zone: 12 nm;
Exclusive economic zone: 200 nm;
Territorial sea: 12 nm
Disputes: claimed by Madagascar
Climate: tropical
Terrain: undetermined
Natural resources: guano, coconuts
Land use: arable land 0%; permanent crops 0%;
          meadows and pastures 0%; forest and woodland 0%;
          other--lush vegetation and coconut palms 100%
Environment: subject to periodic cyclones

Note: located in the Indian Ocean just north of the Mozambique
Channel between Africa and Madagascar

People
Population: uninhabited

Government
Long-form name: none
Type: French possession administered by Commissioner of the
Republic Daniel CONSTANTIN, resident in Reunion

Economy
Overview: no economic activity

Communications
Airports: 1 with runway 1,220-2,439 m
Ports: none; offshore anchorage only


Defense Forces
Note: defense is the responsibility of France
```

Judging by this information, it looks like you have found your ideal home away from home (as long as you can arrange for Internet access). To quit, you press **q** twice: once to stop the paging program and a second time to stop Wais.

Tips for Choosing a Source

Sometimes, the hardest part of using Wais is knowing what sources to use. To help you, here are two tips.

The first tip is to make a preliminary search of the **directory-of-servers** source using keywords that describe the source you think will help you. Remember, though, in this preliminary search, you should be more general than when you are looking for a specific item. Ask yourself, what kind of reference is likely to contain the information I want?

For instance, in our example, we wanted to find a place that was associated with both coconuts and guano, so we needed some type of reference that contained geographical facts about the world. This is why we were successful in searching the **directory-of-servers** with the keywords **geography** and **world**.

Hint

With Wais, being able to think on a higher level of abstraction than your ultimate goal is often the prerequisite to a successful search.

The second tip is a real time saver. You can download a summary of Wais sources by Anonymous FTP. This summary contains short descriptions of all the Wais sources, collected into categories.

A quick and easy way to choose a source is to look in this file, find the category that meets your needs, and use whichever sources look good. Moreover, the file is great for browsing, as it gives you a good overview of everything that Wais can access. The only caveat is that the summary is often out of date; there will probably be newer sources that Wais knows about that are not yet categorized.

You can download the Wais source summary via Anonymous FTP from **kirk.Bond.edu.au**. Look in the directory named **/pub/Bond_Uni/doc/wais** for a file named **src-list.txt**. An alternate Anonymous FTP host is **archive.orst.edu**, where the same file is stored in the directory named **/pub/doc/wais**. (We discuss Anonymous FTP in Chapter 16.)

CHAPTER 24

The World Wide Web

The *World Wide Web*, or more simply, the *Web*, is an ambitious project whose goal is to offer a simple, consistent interface to the vast resources of the Internet.

When you use the Web, you follow your nose: that is, you start anywhere you want, and you jump from one place to another pursuing whatever strikes your fancy. The amazing thing is that with only a few simple commands, you can jump your way around the Internet like a hyperactive flea at a dog convention.

Intrigued? Read on.

What Is the Web?

The World Wide Web has more names than any other Internet resource. We like to call it the Web, but you will also see it referred to as *WWW* or *W3*. To understand the Web, we need to start with the idea of hypertext.

Hypertext is data that contains *links* to other data. A simple example of hypertext is an encyclopedia. Say that you are reading the entry on "Trees". At the end of the article, you see a reference that says, "For related information, see Plants". This last line is a link, from the "Trees" article to the "Plants" article.

Of course, this is a simple example. The Web is based on hypertext that is a lot more complex. In particular, there may be links anywhere within a document, not just at the end.

Here is an imaginary example. Say that you are using the Web to read a hypertext article about trees. Every time the name of a new tree is mentioned there is a link. Each link is marked in some way so that it stands out. For example, a word that has a link may be highlighted or underlined, or it may be identified by a number.

If you follow that link, you will jump to an article about that particular type of tree. Within the main article, there are also links to other related topics such as "rain forests" or "wood". These links lead to complete articles. You will also find links to technical terms such as "deciduous" and "coniferous". When you follow one of these links, you will find a definition.

In the language of the Web, a hypertext *document* is something that contains data and, possibly, links to other documents. The program that you use to read a hypertext document is called a *browser*. As you follow one link to another, we say that you are *navigating* the Web.

In theory, there is no reason to confine ourselves to plain textual data. For example, when we follow a link within the tree document, we might find a picture of a specific type of tree. Or we might find a video clip of an aerial view of a forest. Perhaps we might even find a sound (say, of a tree falling when there is no one around to hear it).

In the world of hypertext, we use the word *hypermedia* to refer to documents that might contain a variety of data types and not just plain text. The Web does contain some hypermedia, but most of what you see will be plain vanilla hypertext: textual material that you can display on your screen.

What makes the Web so powerful is that a link might go to any type of Internet resource: a text file, a Telnet session, a Gopher, a Usenet newsgroup, and so on. The job of your browser is to act as a window into the Internet by following whatever links you desire, and by accessing each document using an appropriate method.

For example, if you follow a link to a text file, your browser will fetch it and display it for you, one screenful at a time. If you follow a link to a Telnet session

(see Chapter 7), your browser will initiate the session for you. You won't have to know anything technical like the address or port number. If you follow a link that leads to a Usenet newsgroup (see Chapter 9), your browser will present the articles using a simple, hypertext format.

Thus, we can characterize the Web a little more precisely. The Web is an attempt to organize all the information on the Internet (plus any local information that you would like to add on your own) as a set of hypertext documents. Although this dream may be somewhat unrealistic, the Web does allow you to access all kinds of Internet resources, just by using a browser to "read" the appropriate document.

Why Would You Use the Web?

The Web was originally developed in Switzerland, at the CERN research center. The idea was to create a way for the CERN physicists to share their work and to use community information. Before long, the idea of the Web expanded and was embraced within the Internet as a general mechanism for accessing information and services.

In theory, the idea of an Internet-wide hypertext system seems to offer the holy grail of information retrieval: a simple, easy method for finding and using just about any type of data that exists. However, in practice, you will find that the Web's usefulness is generally limited to two main functions: reading hypertext articles and accessing Internet resources.

The idea of hypertext is not a new one. Perhaps you have already used such a system. For example, you may have used the Hypercard program on a Macintosh, or the InfoExplorer facility on an IBM RS/6000. Hypertext can be a great way to read information, but there are several unavoidable problems.

First, what makes hypertext more valuable than regular text is having all the links. The problem is that putting links into hypertext is a time-consuming process. It takes so much effort, that there is still not all that much hypertext available. There has been some work done to try to automate the process: to have a program read regular text and try to figure out where to put links. However, there is really no substitute for the judgment of a human being.

The second problem is that the value added to regular text by hypertext links depends on how useful the links really are. Each link is supposed to represent a mental leap that a person would want to make as he or she reads the text. Unfortunately, many types of text do not lend themselves to jumping. Moreover, even when jumps would be useful, it is not always obvious where the links should go. What you personally get out of a hypertext document very much

depends on how close your thinking is to that of the person who put the links in the document.

Still, there are some types of information for which it is more or less obvious where the links should go. For example, a scholarly document might contain a link to each footnote. When you use the Web to read such documents, you will find that hypertext can save you a lot of time. Not only can you jump to a related topic as the need arises, you can also skip a lot of the extraneous details by ignoring the links that look boring (to you, anyway).

The second important use for the Web is accessing Internet resources. The best way for you to see how convenient this can be is to try it for yourself. Indeed, in many cases, the Web will offer you resources that you would never even know existed. We will show you a typical example later in this chapter.

Hint

Set aside a few hours to explore the Web. Once you have a rough idea of what is available, you will be able to call on the Web whenever necessary.

The Web isn't perfect. There will be times when you are better off using a tool that was designed specifically for a particular type of data. For example, although you can use the Web to read Usenet articles, you are usually better off with a regular newsreader. Still, the Web is definitely the most flexible tool you have for exploring the Internet.

Using the Web

Like many Internet resources, the Web uses a client/server system (see Chapter 2). You use a client program called a browser to act as your window into the Web. From the point of view of the Web, everything in the universe consists of either documents or links. Thus, the job of your browser is to read documents and to follow whatever links you select.

A browser knows how to access just about every service and resource on the Internet. It can set up a Wais search, read Usenet articles, access Gopher items, initiate Telnet sessions, and so on. What is most important is that a browser also knows how to connect to *WWW servers* that offer public hypertext documents.

There are dozens of WWW servers around the Internet and most of them specialize in a particular area. For example, the WWW server at the Cornell University Law School contains a great many documents dealing with U.S. law.

In general, you will encounter two types of documents. Text, that can be read, and *indexes*, that can be searched. When your browser finds that a link points to a text document, it retrieves the document and displays it for you, one screenful at a time.

When your browser finds an index, it shows you a short description and invites you to enter one or more keywords. The browser then searches the index and returns whatever data best matches the keywords. A Web search is a lot like a Wais search (see Chapter 23), and, in fact, many indexes are actually Wais sources.

This means that there are only three basic skills you need to use the Web. First, you need to be able to control the display of text. Second, you must know how to tell your browser to follow a link. Third, you must be able to specify how you want to search an index.

The exact way in which these operations work depends on which browser you are using. For example, if your browser uses a graphical user interface with a mouse, the links are highlighted and you can select one simply by clicking on it. With a character-based browser, each link is given a number; to select a link, you enter the number. Still, no matter what type of browser you have, using the Web is easy and intuitive.

Choosing a Browser

More so than most Internet services, your experience working with the Web depends very much on what client program (browser) you use. The best way to access the Web is to use a browser that is designed to take advantage of the special features of your system. For example, if you use X Window (see Chapter 2), you can use an X Window browser that will make use of the mouse and the graphical user interface.

There are browsers available for many different systems: X Window, PCs with Microsoft Windows, Macintosh, VMS, various types of Unix, and so on. It may be that your system already has a browser installed. (You will have to ask around and find out.) If not, perhaps you can prevail upon your system manager to install one. Later in the chapter, we will show you where you can find browser software.

If you do not have a browser on your own computer, you can telnet to one of the hosts listed in Table 24-1. Each of these hosts offers a public browser that can

Location	Internet Address	IP Address
Finland	`info.funet.fi`	`128.214.6.102`
Hungary	`fserv.kfki.hu`	`148.6.0.3`
Israel	`vms.huji.ac.il`	`128.139.4.3`
Slovakia	`sun.uakom.cs`	`192.108.131.11`
Switzerland	`info.cern.ch`	`128.141.201.74`
USA: Kansas	`ukanaix.cc.ukans.edu`	`129.237.1.30`
USA: New Jersey	`www.njit.edu`	`128.235.163.2`
USA: New York	`fatty.law.cornell.edu`	`132.236.108.5`

If you are asked to log in, use a userid of **www**.

Table 24-1. *Public World Wide Web Browsers to Which You Can Telnet*

be used by anyone. With some hosts, the browser will start automatically. With others you must log in as **www**. (We discuss how to use Telnet in Chapter 7.)

All of these hosts offer the type of browser that can be used remotely by people all over the Internet. That means the browser must assume that you are using a plain, ordinary computer terminal with no special features. There are two types of browsers that are used for this purpose. When you telnet to a public Web host, you will be using one of these browsers.

A *line mode browser* is the simplest type. It displays output by writing one line after another to your terminal. To you, it will look as if everything appears at the bottom of your screen and scrolls up. A line mode browser does not care what type of terminal you are using, since all it is doing is sending one line of text after another.

Hint

You may already have a line mode browser on your system. If you do, you may be able to start it by entering the command:

```
www
```

Try it and see what happens.

A *screen mode browser* makes use of your entire screen. Like the line mode browser, a screen mode browser uses only characters. However, it can write those characters anywhere on the screen. This makes the browser faster, and allows it to use a more complex interface. However, it also means that the browser needs to know what type of terminal you are using.

When you log in to a system that uses a screen mode browser, you will be asked to specify your terminal type. Since most systems emulate a VT-100 for remote Telnet sessions, this will be the default.

At the time we wrote this chapter, the hosts in New Jersey, Israel, and Kansas all used (different) screen mode browsers. The other hosts used a simple line mode browser that we will describe in the next section. All of the browsers use English. The Israeli browser also offers Hebrew, while the one in Slovakia also offers Slovak.

An Example of Using the Web

To give you a feeling of what it is like to use the Web, we will show you an example. However, the Web is vast and our example is a small one. To really appreciate how the Web works, you should spend some time exploring it on your own.

In the example, we will telnet to one of the public browsers, the one in Switzerland. We chose this host because it offers a line mode browser (which is easy to use), and because it is oriented towards the general user.

To start, initiate a Telnet session:

```
telnet info.cern.ch
```

Hint

If the line mode browser is already installed on your system, you can use the Web without having to telnet. Just enter the **www** command.

Once you make the connection, the line mode browser starts automatically and you see the welcoming screen:

```
                                                    Overview of the Web (23/27)
                              GENERAL OVERVIEW

There is no "top" to the World-Wide Web. You can look at it from many points
of view. If you have no other bias, here are some places to start:

by Subject[1]          A classification by subject of interest. Incomplete
                       but easiest to use.

by Type[2]             Looking by type of service (access protocol, etc) may
                       allow to find things if you know what you are looking
                       for.

About WWW[3]           About the World-Wide Web global information sharing
                       project

Starting somewhere else

   To use a different default page, perhaps one representing your field of
   interest, see  "customizing your home page"[4].

What happened to CERN?

1-6, Up, <RETURN> for more, Quit, or Help:
```

At the top right of the screen, you see **(23/27)**. This tells you that you are looking at the first 23 lines of a 27-line document. (That is why the last part of the text appears to be cut off.) On the bottom line, you see a reminder of the basic commands that you might want to use. (We will explain them later.)

Although what you see looks simple enough, it is actually hypertext. The numbers in square brackets, such as **[1]** or **[2]**, are the links. To follow one of these links, all you have to do is enter the appropriate number. For example, to jump to the document that explains about the World Wide Web (link **[3]**), all you need to do is type **3** and press RETURN:

```
3
```

Hint

When you use a browser that supports a mouse, you can select a link by clicking on it.

If you want to display the rest of this document (remember, you are looking at only the first 23 lines), you can press RETURN. If you want to display a summary of all the available commands, as well as other useful information, you can enter:

```
help
```

When you are finished working with the Web, you quit by entering:

```
quit
```

You can enter this command at any time, no matter what document you happen to be reading.

Now, to continue with our example, let's say that you want to use the Web to find an interesting fact that most people don't know. Since that's somewhat of a general goal, you can start by exploring.

By looking at the starting document, you see that there are two ways to begin looking for information, by **Subject[1]** or by **Type[2]**. Start with the second choice by entering:

```
2
```

Your browser follows this hypertext link and you now see the beginning of a new document:

```
Data sources classified by access protocol (23/43)
                DATA SOURCES CLASSIFIED BY TYPE OF SERVICE

See also categorization exist by subject[1] .

World-Wide Web[2]       List of W3 servers . See also: about the WWW
                        initiative[3] .

WAIS[4]                 Find WAIS index servers using the directory of
                        servers[5], or lists by name[6] or domain[7].
                        See also: about WAIS[8] .

Network News[9]         Available directly in all www browsers. See also
                        this list of FAQs[10].

Gopher[11]              Campus-wide information systems, etc, listed
                        geographically. See also: about Gopher[12].
```

```
Telnet access[13]        Hypertext  catalogues by Peter Scott. See also:
                         list by Scott Yanoff[14] . Also, Art St George's
                         index[15] (yet to be hyperized) etc.

VAX/VMS HELP[16]         Available using the help gateway[17] to WWW.

1-26, Back, Up, <RETURN> for more, Quit, or Help:
```

Take a look at link **[10]**. It shows you that you can use the Web to access the frequently asked question (FAQ) lists. (We discuss FAQ lists in Chapter 9.) This should be a good place to look for something interesting. Enter:

```
10
```

Your browser follows the link and displays the first part of another document:

```
                    List of USENET FAQs
                      USENET FAQS

This document contains a list of all USENET FAQs found in
news.answers. The document is alphabetized by topic (more or
less). Many of the FAQs in this list are presented in the
same format as they appear in the newsgroup, while others
have been further processed and split into additional documents.
For more information on all aspects of this project, see the
technical notes.[1]

Please send comments and complaints to fine@cis.ohio-state.edu.

   3b1 FAQ[2]
   3b2 FAQ[3]
   Acorn[4]
   Active Newsgroups[5]
   Address Book[6]
   Ai FAQ[7]

1-282, Back, Up, <RETURN> for more, Quit, or Help:
```

Notice the numbers in the bottom left-hand corner: **1-282**. This shows you that there are a total of 282 links in this document (of which you can see **1**

through 7). At this point, there are several commands you can use to examine the rest of the document.

To look at the next screenful of information, press RETURN. To re-display the previous screenful of the current document, enter the command **up**. To start again at the beginning of the document, enter **top**. To go to the end of the document, enter **bottom**.

Hint

Most browser commands can be abbreviated. In the next section, you will find a table that summarizes all the commands and shows the abbreviations.

Remember, your goal is to find something interesting that most people don't know. Press RETURN a few times and look at the next few screenfuls of the document. Eventually, you will see the following:

```
Canadian Football League (CFL) - Frequently Asked Questions [38]
```

Maybe you can find something here. Tell your browser to follow this link, by entering:

```
38
```

This particular link points to an article in the form of a text file, which your browser obligingly fetches. The browser displays the article one screenful at a time. As you read, you can use the same commands that we mentioned earlier: RETURN, **up**, **top**, and **bottom**. So, pressing RETURN, you work your way through the article and come upon the following paragraph:

```
     The Montreal Alouettes folded shortly before the start of the
1987 season, forcing the league to eight teams.  In 1992, the
league announced a plan to first go to twelve, then eventually
sixteen teams, including U.S. teams.  However, these plans were
then reduced to ten, with the two teams being the Sacramento Gold
Miners and a team in San Antonio.  However, only Sacramento will
be joining the league for 1993.
```

Now, here's something interesting that most people don't know: the Canadian Football League has a team in Sacramento (California), with a new team planned for San Antonio (Texas).

Having fulfilled your quota of new knowledge for the day, you decide to quit the Web. Enter:

```
quit
```

The connection is broken and your browser stops.

Commands to Use with the Line Mode Browser

In the previous section, we used the Web's line mode browser for a bit of exploring. In this section, we will go over all the commands, including a few that we have not yet mentioned. If you use a different browser, the commands will, of course, be different. Still, the basic concepts will be similar to what you have already seen.

One nice thing about the line mode browser's commands is that they can all be abbreviated. Most of the commands require only a single letter (such as **h** for **help**); a few require two letters (such as **bo** for **bottom**). The only exception is **quit**, which you must spell out in full. This prevents you from inadvertently aborting your Web session prematurely.

At any time, you can display a quick command summary by using the **help** command. If you would like more detailed assistance, use the **manual** command. This will jump to a hypertext document that contains links you can follow for specific information.

When you enter the **help** command, you will also see some technical information about the current document. For example, if we entered **help** while we were reading the list of FAQs (in the previous section), we would have seen:

```
You are reading
 "List of USENET FAQs"
whose address is
 http://www.cis.ohio-state.edu/hypertext/faq/usenet/top.html
```

The first part of this information reminds us what we are reading. The second part shows us the official technical description of the document's Internet location. This information is called a *Uniform Resource Locator* or *URL*.

Within the Internet, every resource can be described by a URL. We won't go into all the details of URLs, but, for the most part, they are not too hard to understand. The first part of a URL specifies what type of resource is being described, the next part shows the address of the resource.

In this case, the first part is **http**. This indicates we are looking at a hypertext document. (The name **http** refers to Hypertext Transport Protocol, the protocol used by the Web to move data from place to place.) After this, we see the name of a computer at the Computer and Information Services Department of Ohio State University (**www.cis.ohio-state.edu**). Finally, we see a path name on this computer (**hypertext/faq/usenet/top.html**).



FUN TIP: The Web really shows off the power of the Internet. In our example, we use a computer in Switzerland to find a document in the United States (at Ohio State University) that contains information about the Canadian Football League.

If the technical details of the URL don't mean anything to you, don't worry about it. The reason we mention it is because, one day, you may encounter a URL that you want to access using the Web. To do so, you can use the **go** command. Simply type this command, followed by the entire URL.

For example, let's say that one of your friends tells you about a cool document that you should check out. All you have to do is ask him for the URL. You can then use the **go** command to access the document. For example:

```
go http://www.cis.ohio-state.edu/hypertext/faq/usenet/alt-sex/faq/part2/faq.html
```

To continue, there are several commands that you can use to move within a document. To display the next page, just press RETURN. To re-display the previous page, use the **up** command. To move to the beginning of the document, use **top**. To move to the end of the document, use **bottom**.

The next set of commands allow you to use links. The simplest command is to follow a link. To do so, all you need to do is enter its number. For example, to jump to a link shown as **[15]**, enter:

```
15
```

If you want a quick list of all the links in the current document (without any of the text), use the **list** command. This command will also show you the URLs of all the links.

Hint

If there are a lot of links in the current document, the output from the **list** command may scroll by so fast that you cannot read most of it. To slow things down, you will have to use whatever pause key is used with your system. For example, with a Unix system, you can press CTRL-S to pause the output and CTRL-Q to continue. With a PC, you can press the PAUSE key to pause and any other key to continue.

Once you have been exploring for a while, you may want to return to a document that you looked at earlier in the session. Use the **recall** command to display a list of all such documents. Here is a sample list:

```
  Documents you have visited:-

R  1) in Overview of the Web
R  2) in Data sources classified by access protocol
R  3) in List of USENET FAQs
R  4) Canadian Football League (CFL) - Frequently Asked Questions
```

The word "in" means that you have passed through this document to get to where you are.

To jump to one of these documents, use the **recall** command followed by the appropriate number. For example:

```
recall 3
```

There are several other commands that you can use to jump to other documents. To return to the very first document that you saw when you started, use the **home** command. To move back to the previous document in the recall list (the one you just came from), use the **back** command.

Hint

When using the Web, the two most common things to do are:

- enter a number to follow a link forward
- use the **back** command to move one step backward

In other words, entering a number moves you "in"; the **back** command moves you "out". If you use **back** repeatedly, you will follow your path backwards and, eventually, end up at the home document (the very first document you saw).

The next two commands allow you move within a sequence of documents. To jump to the next link from the last document, use **next**. To jump to the previous link from the last document, use **previous**.

At first, these two commands can be a little confusing, so here is an example. Remember that in the last section we looked at a document that had links to FAQ lists. We saw the following list:

```
3b1 FAQ[2]
3b2 FAQ[3]
Acorn[4]
Active Newsgroups[5]
Address Book[6]
Ai FAQ[7]
```

Let's say that you want to jump to the **Acorn** document. Simply enter **4**. While you are reading the **Acorn** document, you can use the **next** command to jump directly to the **Active Newsgroups** document; if you use **next** again, you will jump to the **Address Book** document; and so on.

The **previous** command makes the same type of jump in the opposite direction. For example, if you are reading the **Address Book** document and you use the **previous** command, you will jump to the **Active Newsgroups** document.

The last command we want to mention is used to perform a search. From time to time, you will follow a link that leads to an index. (Remember, "index" is Web terminology for something that can be searched.) Many indexes are actually Wais sources. When you jump to an index, you will see the following on the bottom line of your screen:

```
FIND <keywords>, Back, Up, Quit, or Help:
```

When you see "FIND" it means you are within an index.

You can now type **find**, followed by as many keywords as you want. Your browser will send the search request to the appropriate server and display the results when they arrive. For example, you might enter:

```
find history eighteenth century
```

When you perform a search, there is a shortcut you can use. If the first keyword is not the name of a browser command, you do not have to type **find**. For example, the following two commands would be equivalent:

```
find history eighteenth century
history eighteenth century
```

However, in the following case, you would have to specify **find** as the first keyword happens to be the name of a command:

```
find top performer
```

For reference, Table 24-2 contains a summary of all the basic commands that we discussed in this section.

Finding a Browser

We mentioned earlier that, if you get serious about the Web, you should definitely use a browser on your own computer. If one is not already installed, ask your system manager if he or she could install one for everybody.

There are browser programs for a wide variety of different systems available via Anonymous FTP from **info.cern.ch**. For more information, look in the directory named **/pub/www** for a file named **README.txt**.

If you have an X Window system, we recommend that you try the Mosaic browser. Mosaic was written by Marc Andreessen of the National Center for Supercomputer Applications (NCSA) at the University of Illinois at Urbana.

Mosaic has a large number of advanced features, such as embedded graphics and the launching of multimedia applications. It also uses colors to help you keep track of links. For example, Mosaic remembers every document you have ever read. When a link points to something you have already seen, Mosaic will draw that link using a special color. This helps you to avoid wasting your time

Command	Full Name	Description
–	`number`	follow the specified link
`b`	`back`	jump to previous document in recall list
`bo`	`bottom`	jump to the end of the current document
`f` *words*	`find` *words*	use specified keywords to search index
`g` *UDL*	`go` *UDL*	jump to specified document or resource
`h`	`help`	display command summary and technical info
`ho`	`home`	jump to the starting document
`l`	`list`	display a list of links within current document
`m`	`manual`	jump to the browser reference document
`n`	`next`	jump to the next link within the last document
`p`	`previous`	jump to previous link within the last document
–	`quit`	quit the Web
`r`	`recall`	display list of previously-visited documents
`r` *number*	`recall` *number*	jump to specified document from recall list
–	RETURN	display next screenful of the current document
`t`	`top`	jump to the beginning of the current document
`u`	`up`	display previous screenful of current document

Table 24-2. *Summary of Line Mode Browser Commands*

on documents you have already seen. It also makes it easy to follow your previous path through unknown territory.

Another nice feature is that Mosaic allows you to attach personal notes within any document. These notes will appear each time you read the document. (Of course, the notes are private; only you can see them.)

Mosaic looks and feels like a Motif application, although you don't need to have Motif on your computer. (Motif is a graphical user interface that runs on top of X Window.)

Installing Mosaic is easy, as it comes all ready to run. All you need to do is download the program, uncompress it (using the **uncompress** command) and make it executable (using the **chmod** command). There is nothing to set up.

Mosaic is available via Anonymous FTP from **ftp.ncsa.uiuc.edu**. The actual name of the program is **xmosaic**. Change to the directory **/Web/xmosaic-binaries** and look for the version for your system.

If you get interested in the World Wide Web, there is a Usenet newsgroup that you can subscribe to called **comp.infosystems.www**. For a discussion of hypertext, subscribe to **alt.hypertext**.

(We discuss X Window in Chapter 2, Anonymous FTP in Chapter 16, uncompressing files in Chapter 18, and Usenet in Chapter 9.)



CHAPTER 25

Mailing Lists

In this chapter, we will show you how to access a huge network of discussion groups: one that is carried entirely by mail. To participate in these discussions, you do not need to use Usenet or any other Internet services. All you need to know is how to send and read mail (which we discuss in Chapters 5 and 6).

There are thousands of discussions going on right now on every topic imaginable. And to participate, all you need is an electronic mail address.

515

What Is a Mailing List?

Like all successful Internet services, mailing lists are based on a simple idea. When you mail a message to someone, you specify an address. If you want to mail a message to more than one person, you can set up a special name, called an *alias*, that represents a group of people.

For example, let's say you set up an alias called **executives** to represent the addresses of three people named Curly, Larry, and Moe. Whenever you mail a message to **executives**, the mail program will automatically send it to each of these three users.

Imagine how these three people could use this alias to have a discussion group. Say that Curly gets an idea that he wants to share with the others. All he has to do is mail a message to **executives** and everybody gets a copy. Now let's say that Moe wants to comment on something in Curly's message. He sends his own message to **executives**. Again, the message is automatically sent to everybody on the list.

Now, think about the same sort of thing on a larger scale. Imagine an alias that contains the mail addresses of tens or even hundreds of users, scattered all around the Internet. Any message sent to the alias will be automatically sent to everyone in the group. People can talk, argue, help one another, discuss problems, share information, and so on. Everything that anyone says goes to all the people in the group.

This is a *mailing list*. As an Internet user, you have access to several thousand such lists, each of which is devoted to a specific topic.

How Are Mailing Lists Different from Usenet Newsgroups?

In Chapters 9 and 10, we discussed Usenet, the worldwide collection of discussion groups. (In Usenet, we refer to the discussion groups as newsgroups even though they do not contain actual news.) We mentioned that there are over 5,000 different Usenet newsgroups.

Many of these newsgroups are of local or regional interest, and even large computer systems will usually carry no more than a couple of thousand newsgroups. Still, that is a huge variety. To see for yourself, take a moment to glance at Appendix G, which contains descriptions of many of these newsgroups.

Since Usenet offers such a large variety of discussion topics, it is natural to ask how Usenet newsgroups differ from mailing lists.

The first big difference is that you have to learn a lot more to participate in Usenet. In particular, you have to learn how to use a newsreader program. Such programs are complex, and it may take a while for you to feel at home with Usenet. With a mailing list, all you need to know is how to send and read mail, something which you should probably learn anyway. This means that people who do not have access to Usenet—or don't want to learn how to use it—can still participate in discussion groups.

The next difference is that the discussions in a mailing list come to you in the form of messages sent to your personal mailbox. Some lists have only a few messages a day. However, it is not uncommon for a busy list to generate dozens of messages a day. Since all these messages show up in your mailbox, you have to do something with them. Usenet articles, on the other hand, are stored in a central location on your network and are administered by a system manager. When you participate in mailing lists, it is not uncommon to return from a two week vacation and find hundreds of messages in your mailbox. (However, as we will explain later in the chapter, it is usually possible to tell a mailing list to stop sending you messages temporarily.)

Hint

Mailing lists can generate a lot of messages. Although you can subscribe to as many mailing lists as you want, it is best to confine yourself to no more than five. Otherwise, you are guaranteed to find your mailbox constantly filled with unread messages.

One of the nice things about mailing lists is that you can choose a few that interest you and count on getting the messages automatically. There is nothing for you to do but read your mail. With Usenet, you have to start your newsreader program and check your favorite newsgroups every time you want to see what has arrived. Moreover, most system managers automatically delete Usenet articles after they have been around for a fixed period (anywhere from 1-2 days to several weeks).

However, Usenet is more convenient in other ways. As a Usenet user, you only need to participate when you want. This means that you can drop in and out of discussions as the mood strikes you. Moreover, it is easy to sample a variety of newsgroups quickly. With a mailing list (as you will see), you have to send a special mail message to be put on the list and another message to be taken off.



FUN TIP: The best part about both Usenet and mailing lists is that they are free.

Hint

To use a mailing list, there are two main ideas you need to understand. First, you should understand the basic concepts behind the Internet mail system and how to use a mail program. Second, you need to know how to read and understand Internet addresses.

If you feel like you need a quick review, you might want to take a few moments to skim Chapters 4 and 5 where we discuss these topics. If you use the standard Unix mail program, you can also read Chapter 6. Otherwise, you will have to find some documentation (or a friendly person) to help you with whatever mail program you are using.

Moderated and Unmoderated Mailing Lists

Each mailing list has an administrator. In most cases, this is one person, variously referred to as the list manager, administrator, or coordinator. The main job of the administrator is to keep the list of mail addresses up to date.

All mailing lists have an official address. With most lists, every message that is sent to that address is automatically passed on to everyone on the list. This means that anyone can contribute to a mailing list just by sending a message to the appropriate address.

When you receive a message, it will be sent to you from that same address. Thus, if you use the built-in feature of your mailing program to reply to the sender, your reply will go to everyone on this list.

Hint

Before you reply to any message, always ask yourself if it would be better to send a private response to the person who sent the message. (This person's name will be in the message.) In many cases, it is more appropriate to send a private response than a message that will be sent to everyone in the group.

Some mailing lists are *moderated*. This means that all the messages go to one person called the *moderator* (which may or may not be the same person as the administrator). The moderator decides which articles should be sent out to the members of the list. Most moderators also perform some basic editing and organization on the raw material.

Some moderators will organize messages into a collection called a *digest*. A digest is like an issue of an electronic magazine: a whole set of messages and articles in one easy to read package. Some moderators will include a table of contents with each digest so it is easy to find the messages that interest you.

The advantage of a moderated mailing list is that you see only the best messages (in someone's opinion, anyway). Many unmoderated mailing lists have a lot of boring and redundant messages that you will have to wade through to find the jewels.

The main disadvantage of a moderated list is that maintaining it is a lot of work. Moreover, the only compensation that moderators receive is that warm feeling that comes to those who help their fellows. (Thus, most mailing lists are not moderated.)

How Mailing Lists Are Administered

As we explained in the last section, the most important part of administering a mailing list is keeping track of the people on the list. When you request to be put on a mailing list, we say that you *subscribe* to that list. When you ask to be taken off the list, you *unsubscribe*. (Remember, though, even though we talk about "subscribing", there is no charge for the service.)

There are two basic ways in which mailing lists are administered. Some lists are maintained by a program. To subscribe, you send a message to a special address. All mail to this address is automatically processed by the list administration program. Since the message will be read by a program, you must use a particular format (which we explain later).

Other lists are maintained by a person. With these lists, you send a message that is read by the administrator, who manually adds or deletes people from the list. Although most lists are public, there are some private lists that you cannot join without the permission of the administrator.

The most common mailing list administration system is called *Listserv*. (The name stands for "list server".) The convention is to speak of a computer that provides this service as "a Listserv". For example, you might read, "To get basic information about subscribing to a mailing list, send a message with the word **help** to any Listserv."

The Listserv system was developed to coordinate mailing lists on the large Bitnet network (described in the next section). We will discuss Listserv and the commands that it uses later in the chapter.

Bitnet

Bitnet is a worldwide network—separate from the Internet—that connects well over a thousand academic and research institutions in more than 40 countries. Many of the Bitnet sites are IBM mainframe computers running the VM operating system and supporting hundreds of users. Thus, Bitnet serves a very large number of people. In this section, we will explain a little about Bitnet because, as you will see, it is the source of many of the mailing lists to which you have access.

Bitnet began in 1981 as a small network of IBM mainframe computers at the City University of New York (CUNY). The name Bitnet was chosen to stand for the "Because It's Time Network". (No doubt, this name would make sense to you if you were using an IBM mainframe computer in New York in 1981.)

In the United States, Bitnet is associated with Educom, a non-profit consortium of educational institutions, and is administered by the Corporation for Research and Educational Networking (CREN). Educom supports the Bitnet Network Information Center (Bitnic). The job of Bitnic is to promote the use of Bitnet in higher education.

Outside the U.S., Bitnet is known by different names. In Canada, it is called Netnorth. In Europe, it is the European Academic Research Network (EARN). In Latin America and Asia, you will find other names for Bitnet.

In Chapter 2, we discussed protocols (technical specifications) and explained how the Internet is based on a family of protocols called TCP/IP. Bitnet is based on a family of IBM protocols called RSCS (Remote Spooling Communications Subsystem) and NJE (Network Job Entry).

Historically, Bitnet developed within a technology that did not allow for systems like Usenet. Thus, an elaborate mailing list system, based on Listserv, developed.

Hint

If you hang around Bitnet, you will see many common abbreviations in the names of computers and mailing lists: **vm** (the name of the principle IBM mainframe operating system), **cuny** (City University of New York, where Bitnet was developed), **bitnic** (Bitnet Network Information Center, a main site of Bitnet in the U.S.), **earn** (European Academic and Research Network, the European part of Bitnet), and so on.

You will also find that Bitnet people tend to use acronyms and commands that are all uppercase letters, like BITNET and LISTSERV. This is part of the IBM mainframe culture. Compare this to our more genteel Internet traditions, which encourage the use of terminology that is almost exclusively lowercase.



FUN TIP: You, can learn a lot about a culture by observing how it uses upper- and lowercase letters.

Bitnet Mailing Lists and Usenet

Many (but not all) of the Bitnet lists are available via Usenet as well as through the mail. Each message that is sent to one of these lists is also sent to a Usenet newsgroup which you can read in the regular fashion. All these Bitnet newsgroups are in the **bit** hierarchy and have names that start with **bit.listserv**. (We discuss newsgroup names in Chapter 9.)

For example, the Bitnet mailing list named **film-l** is devoted to film and the cinema. Messages to this list can also be read as Usenet articles in the newsgroup **bit.listserv.film-l**. Moreover, when you send an article to the Usenet group, it will be forwarded to the list itself and, from there, to all the subscribers. (Although, this doesn't always work properly.)

If you would like to see a list of the Usenet newsgroups that contain Bitnet mailing lists, look at Appendix G and read the descriptions of the newsgroups in the **bit** hierarchy.

Hint

Many Bitnet mailing lists have names that end in **-l** (the "l" stands for list).

The actual work of sending messages between Usenet and Bitnet is done by a computer called an *Bitnet/Usenet gateway*. There are a number of such computers that act as gateways for the various lists. If you would like more information about the gateways or about the Bitnet/Usenet newsgroups, there are three articles you can read.

The first article, named **bitlist**, contains a master list of all the Bitnet-oriented newsgroups, with a one-line description of each group. The second article, named **gatelist**, shows each Bitnet mailing list, its corresponding newsgroup, and the name of the gateway. The third article is named **policy**. It provides a general explanation of Usenet, Bitnet, and how they are connected.

All three articles are posted regularly to the newsgroups **bit.admin** and **news.answers**. They are also available via Anonymous FTP from **rtfm.mit.edu** in the directory **/pub/usenet/news.answers/bit**.

Sending Mail to Bitnet

To participate in a mailing list, all you have to do is find the list and send a message asking to subscribe. We will explain how to do all of that later in the chapter. First, though, we need to take a moment to discuss how to mail a message to Bitnet. That is because many of the mailing lists that you will encounter are Bitnet lists. To subscribe (and unsubscribe), you will need to send a message to a Listserv. And once you subscribe, you will need to send messages to the list itself in order to participate. In both cases, you may find yourself having to deal with a Bitnet address.

Before we start, we would like to remind you that we discuss Internet addresses in Chapter 4. At the end of that chapter, we explain how to mail a message to Bitnet. You might want to take a moment to review that material before you go on.

Within Bitnet, addresses are simple. All you need is a user name and a computer name. For example, say that a Bitnet user has a friend whose user name is **lunaea** on the computer known as **psuvm**. He or she could mail a message to the friend by using the address **lunaea@psuvm** (or as a Bitnet person would put it, **LUNAEA@PSUVM**).

From the Internet, we cannot send mail directly to Bitnet. It must first go through a *Bitnet/Internet gateway*: a computer that is connected to both networks. If we wanted to send a message to the same person we mentioned above, we would have to send a message to a gateway and tell it: "Please take this message and pass it to the Bitnet computer named **psuvm** and, once there, have that computer deliver the message to **lunaea**."

On many Internet systems, this is easy. All you have to do is use the following address format:

```
username@computer.bitnet
```

For example:

```
lunaea@psuvm.bitnet
```

What happens is that the mail program on your system recognizes the name **bitnet** (called a pseudo domain) and knows how to send your message to a Bitnet/Internet gateway.

Some Internet systems, however, are not set up to recognize the **bitnet** pseudo domain. If your system is like this, your mail will come back marked "Host unknown" (like in the Elvis Presley song). In such cases, you will have to send your Bitnet messages directly to a gateway. These days, many Bitnet computers are also on the Internet and can act as gateways between the two networks. For reference, the list below shows a number of such gateways that you can use.

cornellc.cit.cornell.edu
cunyvm.cuny.edu
mitvma.mit.edu
pucc.princeton.edu
vm1.nodak.edu

When you send mail via a gateway, there are two addressing formats that you can use. The preferable one uses the UUCP bang path notation (that we discussed in Chapter 4):

```
gateway!computer.bitnet!userid
```

For example, to send a message to **lunaea** on the Bitnet computer named **psuvm**, you might use the following address:

```
cornellc.cit.cornell.edu!psuvm.bitnet!lunaea
```

Hint

If you are using a Unix system with the C-Shell, the ! character will have a special meaning. Consequently, when you enter an address that contains ! characters, you will see the following error message:

```
Event not found.
```

If this happens, you must place a \ (backslash) character in front of each ! character. For example:

```
cornellc.cit.cornell.edu\!psuvm.bitnet\!lunaea
```

The \ character tells the C-Shell that the next character is to be taken literally.

The second format uses the % notation (also discussed in Chapter 4):

```
userid%computer.bitnet@gateway
```

For example:

```
lunaea%psuvm.bitnet@cornellc.cit.cornell.edu
```

Hint

When sending mail via a Bitnet/Internet gateway, it is better to use the bang path format. The % format is not officially supported anymore (although it usually works). The bang path format will always work and is easier to understand.

Here are two real life examples of how to mail messages to Bitnet. To send a message to a Listserv, you use a user name of **listserv**. Thus, here are three addresses that you might use to send a message to the Listserv at the Bitnet computer named **templevm**:

```
listserv@templevm.bitnet
cunyvm.cuny.edu!templevm.bitnet!listserv
listserv%templevm.bitnet@cunyvm.cuny.edu
```

In the earlier examples, we used the gateway at Cornell University. In these examples, we use the gateway at CUNY. It really doesn't matter which one you use, although you may get faster service if you use the gateway closest to you.

Here is our second real-life example. Most Bitnet lists are administered by a single Bitnet computer. When you read a description of a mailing list, you will typically see the name of the list and the name of the computer. For example, the list named **help-net** resides at **templevm**. When you read about it, it will be called **help-net@templevm**. (By the way, the **help-net** list is devoted to general questions and answers regarding Bitnet and the Internet.)

Let's say that you have already subscribed to this list. To mail a message to the list, you will have to send it to user name **help-net** at the computer **templevm**. As an Internet user, here are three addresses you could use:

```
help-net@templevm.bitnet
cunyvm.cuny.edu!templevm.bitnet!help-net
help-net%templevm.bitnet@cunyvm.cuny.edu
```

When a Bitnet computer is also on the Internet, it will have an Internet address as well as a Bitnet address. In such cases, you can use the Internet address (which will be faster). For example, if you wanted to send mail to the Listserv at CUNY, you could send it directly to:

```
listserv@cunyvm.cuny.edu
```

The last thing we would like to mention is that some lists can be administered by more than one Bitnet computer. These computers are called *peers*. In such a case, you can send mail regarding the list to any one of the peers.

Finding Mailing Lists

By now, you are probably wondering what mailing lists are available. Well, there are a lot of them, and they cover just about any topic you can imagine. Of course, there are new lists being formed all the time, and (less often) old lists disappearing.

If you want to find out what is available, there are a number of sources from which you can obtain a summary of mailing lists. Such a summary is usually referred to as a *list-of-lists*. In this section, we will show you where to obtain various lists-of-lists. Each one is put together by different people (usually volunteers) and covers a different territory.

> **Hint**
>
> Most of the lists-of-lists are large and take up a lot of disk space. If you copy a list-of-lists to your computer, we suggest that you delete the file when you are finished looking at it. There is no real reason to save a list-of-lists: it would just go out of date and you can get a newer version for free whenever you want.

The SRI List-of-Lists

One of the two largest list-of-lists is maintained by SRI International at their Network Information Systems Center (NISC) in Menlo Park, California. The purpose of this list-of-lists is to offer a description of every public mailing list—both Internet and Bitnet—along with instructions on how to subscribe.

This list is a useful, albeit very long reference. Unfortunately, the instructions are somewhat confusing because they were written by different people at different times. For example, there is really only one way to subscribe to a Bitnet list (as you will see later in the chapter). However, in the SRI list-of-lists, each Bitnet list has a separate description that shows you how to subscribe. Actually, all you really need to do is find out where a mailing list is, and then follow the instructions in this chapter.

There are two ways to get the SRI list-of-lists. First, you can download it (copy it to your computer) via Anonymous FTP from **ftp.nisc.sri.com**. Look in the directory named **/netinfo** for a file named **interest-groups.Z**. You will have to uncompress this file once you download it. (We discuss Anonymous FTP in Chapter 16. We discuss uncompressing files in Chapter 18.)

The second way to get this list is to send mail to **mail-server@nisc.sri.com**. In the body of the letter, put the message **send /netinfo/interest-groups**. (Note: The list is huge. It will arrive in the form of many separate messages that you will have to save and join together.)

The Dartmouth List-of-Lists

The other very large list-of-lists is maintained at Dartmouth University in New Hampshire. Like the SRI list, the Dartmouth list also attempts to keep track of every Internet and Bitnet mailing list in existence. The Dartmouth people edit their list to remove duplications and mailing lists of only local interest.

Before you download this list (which comes in parts), you should read a file that contains instructions. Use Anonymous FTP to connect to **dartcms1.dartmouth.edu**. This system is an IBM mainframe and it may look a little strange if you are used to Unix computers; still, the regular FTP commands will work.

Change to the directory named **siglists** (use **cd siglists**). Download the instruction file named **read.me** (use **get read.me**). Within this file, you will see several ways to obtain the list-of-lists. If you want to download it using regular FTP, use the command **get listserv.lists**.

Wais

When you are interested in a particular subject and you want to find a mailing list in that area, it is a bother to have to download and examine a large set of files. An easier way is to use Wais. All you have to do is search the source (Wais database) named **mailing-lists**. (We explain how to use Wais in Chapter 23.) This source contains several Bitnet and Internet lists-of-lists as well as a list of Usenet newsgroups.

Listserv Mailing Lists

You can get a summary of all the Listserv mailing lists on Bitnet by mailing a message to any Listserv. In the body of the message, put the command **list global**. The Listserv will mail you back a list in which each mailing list is summarized on a single line.

If you are interested in a particular subject, use the command **list global/*subject***, specifying whatever subject you want. For example, if you want a

summary of all the lists that have something to do with poetry, use **list global/poetry**.

Here is an example of a message that requests the overall list. Notice that when you send commands to a Listserv you do not need to specify a Subject. (Remember, you can send this request to any Listserv.)

```
mail listserv@cunyvm.cuny.edu
Subject:
list global
```

The Internet List-of-Lists

There is a list-of-lists that keeps track of public Internet mailing lists. This list is updated regularly and posted to the Usenet newsgroups **news.lists**, **news.announce.newusers**, and **news.answers**.

You can download the list via Anonymous FTP from **rtfm.mit.edu**. Look for the files named **part1**, **part2**, and so on, in the directory **/pub/usenet/news.answers/mail/mailing-lists**.

Academic Mailing Lists for Scholars

A group at Kent State University in Ohio maintains a list-of-lists that describes mailing lists devoted to scholarly and academic subjects. If you are a student or teacher, this is a good place to look for a mailing list for your discipline.

You can download this list-of-lists via Anonymous FTP from **ksuvxa.kent.edu**. Once you log in, change to the **library** directory. (Use the command **cd library**.) For instructions and a description of the list-of-lists, download the file named **acadlist.readme**. The actual list is divided into parts and stored in the files named **acadlist.file1**, **acadlist.file2**, and so on.

Reviews of Mailing Lists

If you are interested in reviews of mailing lists, you can subscribe to the Bitnet mailing list named **lstrev-l@umslvma**. This is a service provided by Raleigh C. Muns. He subscribes to a mailing list for a while, then offers his comments to the world.

New Bitnet Mailing Lists

There is a Bitnet mailing list that announces all new lists. You can subscribe to this list in order to keep up on what is new and exciting. The name of the mailing list is **new-list@vm1.nodak.edu**. This list is also available on Usenet as the newsgroup **bit.listserv.new-list**.

Subscribing to a Bitnet Mailing List

To subscribe to a mailing list, all you have to do is mail a simple message. If the list is administered by a program, such as Listserv, you send the message to that program. If the list is administered by a person, you send a message to that person.

When you subscribe to a mailing list, there are only two important guidelines to remember. First, you must make sure that you send your requests to the address of the person or program that administers the mailing list. Do not send requests to the address of the list itself. Second, when you send a message to a program, you must use commands in a particular format.

> **Hint**
>
> Remember, many Bitnet mailing lists are also available on Usenet.

We can divide the whole world of mailing lists into two groups: Listserv-administered lists and manually administered Internet lists. When you subscribe to a mailing list, the first thing you have to do is figure out if it is administered by a Listserv. You can usually do so by reading the description of the list.

If you are dealing with a Listserv-administered list, there will be several clues. First, the description may tell you outright that to subscribe you must send a message to a particular Listserv. Second, the description may tell you that this is a Bitnet mailing list. (Virtually all Bitnet lists use Listserv.) Third, you may see instructions telling you to send a subscription message to an address that starts with the name **listserv**. Finally, you may recognize that address of the list as a Bitnet address.

Here is a typical example. Let's say you see the following description of a mailing list:

```
HELP-NET on LISTSERV@TEMPLEVM.Bitnet

Help-Net is a discussion list for the purposes of solving user
problems with utilities and software related to the Internet and
Bitnet networks.  In addition, LISTSERV at TEMPLEVM maintains a
set of low-level help files intended to help the beginning user
acclimatize himself to the network systems.
```

You can tell that you would use Listserv to subscribe to this mailing list: the top line contains the address of a Listserv. To subscribe to the mailing list, you would send mail to the Bitnet computer named **templevm**. Use the address:



```
listserv@templevm.bitnet
```

(If this address doesn't work, you can use one of the variations we discussed earlier.)

The mailing list itself will have an address at the same computer as the Listserv. Thus, once you have subscribed to the list, you can participate by sending mail to:



```
help-net@templevm.bitnet
```

Sometimes you will see a much briefer description of a mailing list. For example:

```
HELP-NET          HELP-NET@TEMPLEVM Bitnet/Internet Help Resource
```

In this case, all you really see is the address of the list itself, **help-net@templevm**. You have to be able to look at this address and realize that it refers to a Bitnet computer. (You can tell because the address looks like a Bitnet address and not like an Internet address.) Once you realize that you are dealing with a Bitnet mailing list, you know that you can subscribe by sending a message to the Listserv at the same address.

To subscribe to a Bitnet mailing list, mail a short message to the Listserv at the same address. You do not need to put a subject in the message. In the body of the message, put a single line that contains the word **subscribe**, followed by the name of the mailing list, followed by your full name. The following is a typical subscription message:

```
mail listserv@templevm.bitnet
Subject:
subscribe help-net Chuck Wagon
```

This message asks the Listserv at **templevm** to add the person named Chuck Wagon to the **help-net** mailing list. Notice that you do not have to specify your mailing address: the Listserv will figure it out from the header of your message.

To unsubscribe to a Bitnet mailing list, send a message to the Listserv that administers the list. In the body of the message, put a single line that contains the word **unsubscribe** followed by the name of the mailing list. For example:

```
mail listserv@templevm.bitnet
Subject:
unsubscribe help-net
```

Hint

What do you do if you accidentally spelled your name wrong when you subscribed to a Bitnet mailing list? Not to worry. Simply send in another **subscribe** message with your name spelled correctly. You do not need to unsubscribe just to change your name.

Aside from **subscribe** and **unsubscribe**, there is a whole set of commands that you can send to a Listserv. All you have to do is send a message to the Listserv, with a single line that contains the command. The Listserv will read your message, carry out the command, and mail you the results.

The first command you might want to use is **help**. This will send you a summary of all the basic commands. You can also use the **info** command to request information on a particular topic. Use **info ?** to ask for a list of topics.

One interesting command is **review**. Type this command followed by the name of a mailing list. For example, to ask for information about the **help-net** mailing list, use:

```
review help-net
```

The Listserv will send you information about the list, including a description of the list, and the names and addresses of all the subscribers.

The Bitnet Listservs have an interesting characteristic. Whenever you send a request for information, the Listserv will send you two messages. The first is a confirmation message; the second is the actual information. Here is a typical confirmation message that we received after sending the command **list global** to a Listserv:

```
From: BITNET list server at BITNIC (1.7f)
<LISTSERV@BITNIC.EDUCOM.EDU>
Subject:      Output of your job "harley"

> list global

Summary of resource utilization
-------------------------------

 CPU time:         4.997 sec                Device I/O:    158
 Overhead CPU:     1.358 sec                Paging I/O:    213
 CPU model:         9377                    DASD model:   9335
```

The name **harley** that you see in the **Subject:** line is the userid that sent the mail to the Listserv. This name is inserted automatically.

A short time after we received this message, another message arrived with the information we requested. You will receive similar confirmation messages whenever you subscribe or unsubscribe to a mailing list, and (if you choose) whenever you send a message to a list.

To control certain aspects of a mailing list subscription, you can use the **set** command. This command tells the Listserv that administers a mailing list that you request certain options regarding that list. Here is how it works.

Send a message to the appropriate Listserv. Within the body of the message, put a single line that contains **set**, followed by the name of the mailing list, followed by the option you want. For example:

```
set help-net noack
```

This command tells the Listserv to set the **noack** option (explained below) for you with respect to the **help-net** mailing list. There are a number of options you can use, of which we will discuss the most important.

To tell the Listserv that you do not want to receive confirmation messages whenever you send mail to a list, use **noack** (no acknowledgment). To turn this service back on, use **ack**. The default is **noack**.

If you do not want to receive mail temporarily (say, while you are away on vacation), you can use the **nomail** option to tell the Listserv to stop sending you **mail**. When you want to start receiving mail again, use the **mail** option. When

you use **nomail**, you are still subscribed to the mailing list. Thus, if you want to forget about the list completely, you should unsubscribe.

When you send a message to a mailing list, the Listserv that administers the list does not send you a copy of the message. If you would like to receive copies of your own messages, you can use the **set** command with the **repro** option. To turn this off again, use the **norepro** option.

For reference, Table 25-1 contains a summary of these basic Listserv commands. Notice that **signup** is a synonym for **subscribe**, and **signoff** is a synonym for **unsubscribe**.

Subscribing to an Internet Mailing List

You will find that some Internet mailing lists are administered by Listserv-like software running on an Internet computer. Still, from our point of view, they

Command	Function
`help`	send summary of basic commands
`info ?`	send list of information topics
`info` *`topic`*	send information about specified topic
`list global`	send summary of all Bitnet mailing lists
`review` *`list`*	send information about specified mailing list
`set` *`list`* `ack`	send confirmation messages for specified list
`set` *`list`* `noack`	do not send confirmation messages for specified list
`set` *`list`* `mail`	start sending mail from specified list
`set` *`list`* `no mail`	stop sending mail from specified list
`set` *`list`* `repro`	send copies of your own messages
`set` *`list`* `norepro`	do not send copies of your own messages
`subscribe` *`list yourname`*	subscribe to specified mailing list
`signup` *`list yourname`*	same as `subscribe`
`unsubscribe` *`list`*	unsubscribe to specified mailing list
`signoff` *`list`*	same as `unsubscribe`

Table 25-1. *Summary of Important Listserv Commands*