In this chapter:

- *Remote Logins*
- *Windows from Other Computers*
- *Lynx, a Text-based Web Browser*
- *Transferring Files*
- *Electronic Mail*
- *Usenet News*
- *Interactive Chat*

6

Using the Internet and Other Networks

A network lets computers communicate with each other, sharing files, email, and much more. Unix systems have been networked for more than 25 years.

This chapter introduces Unix networking: running programs on other computers, copying files between computers, browsing the World Wide Web, sending and receiving email messages, reading and posting messages to Usenet "Net news" discussions, and "chatting" interactively with other users on your local computer or worldwide.

Remote Logins

The computer you log in to may not be the computer you need to use. For instance, you might have a workstation on your desk but need to do some work on the main computer in another building. Or you might be a professor doing research with a computer at another university. Your Unix system can connect to another computer to let you work as if you were sitting at that computer. This section describes how to connect to another computer from a local terminal. If you need to use a graphical (nonterminal) program, the section "Windows from Other Computers," next, explains.

To log into a remote computer using a terminal, first log in to your local computer (as explained in the section "Logging in Nongraphically" in Chapter 1, or in the section "A. Ready to Run X (with a Graphical Login)" in Chapter 2). Then, in a terminal or terminal window on your local computer, start a program that connects to the remote computer. Some typical

programs for connecting over a computer network are **telnet**, **ssh** ("secure shell"), **rsh**, ("remote shell") or **rlogin** ("remote login"). Programs such as **cu** and **tip** connect through telephone lines using a modem. In any case, when you log off the remote computer, the remote login program quits and you get another shell prompt from your local computer.

Figure 6-1 shows how remote login programs such as **telnet** work. In a local login, you interact directly with the shell program running on your local system. In a remote login, you run a remote-access program on your local system; that program lets you interact with a shell program on the remote system.

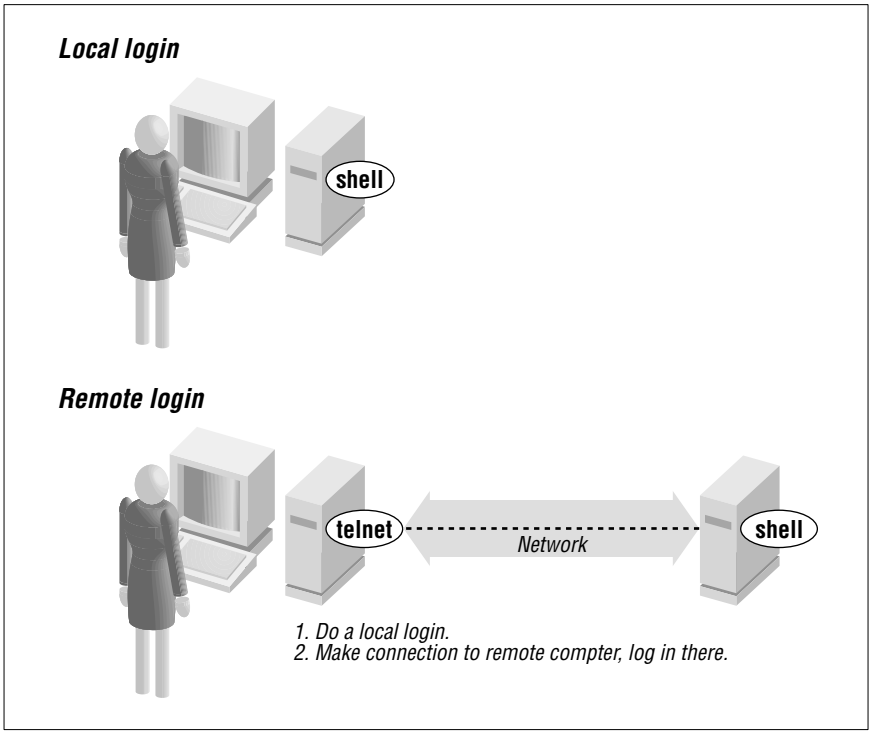

Figure 6-1. Local login, remote login

The syntax for most remote login programs is:

program-name remote-hostname

For example, when Dr. Nelson wants to connect to the remote computer named *biolab.medu.edu*, she'd first make a local login to her computer named *fuzzy*. Next, she'd use the **telnet** program to reach the remote computer. Her session would look something like this:

```
login: jennifer
Password:

NOTICE to all second-floor MDs: meeting in room 304 at 4 PM.

fuzzy$ telnet biolab.medu.edu

Medical University Biology Laboratory

biolab.medu.edu login: jdnelson
Password:

biolab$
        .
        .
        .
biolab$ exit
Connection closed by foreign host.
fuzzy$
```

Her accounts have shell prompts that include the hostname. This reminds her when she's logged in remotely. If you use more than one system but don't have the hostname in your prompt, see the section "Documentation" in Chapter 8 to find out how to add it.

When you're logged on to a remote system, keep in mind that the commands you type will take effect on the remote system, not your local one! For instance, if you use **lpr** or **lp** to print a file, the printer it comes out of may be very far away.

The programs **rsh** (also called **rlogin**) and **ssh** generally don't give you a "login:" prompt. These programs assume that your remote username is the same as your local username. If they're different, give your remote username on the command line of the remote login program, as shown in the next example.

You may be able to log in without typing your remote password or passphrase.* Otherwise, you'll be prompted after entering the command line.

* In **ssh**, you can run an *agent* program, such as **ssh-agent**, that asks for your passphrase once, and then handles authentication every time you run **ssh** or **scp** afterward. For **rsh** and **rcp**, you can either store your remote password in a file named *.rhosts* in your local home directory, or the remote system can list your local computer in a file named *hosts.equiv* that's set up by the system administrator.

Following are four sample **ssh** and **rsh** command lines. (You may need to substitute **rlogin** for **rsh**.) The first pair show the way to log in to the remote system, *biolab.medu.edu*, when your username is the same on both the local and remote systems. The second pair show how to log in if your remote username is different (in this case, *jdnelson*); note that your version of **ssh** and **rsh** may support both syntaxes shown:

```
$ ssh biolab.medu.edu
$ rsh biolab.medu.edu
$ ssh jdnelson@biolab.medu.edu
$ rsh -l jdnelson biolab.medu.edu
```

About Security

Today's Internet, and other public networks, have users (called *crackers*; also erroneously called *hackers*) who try to break into computers and snoop on other network users. Most remote login programs (and file transfer programs, which we cover later in this chapter) were designed 20 years ago or more, when networks were friendly places with cooperative users. Those programs (many versions of **telnet** and **rsh**, for instance) make a cracker's job easy. They transmit your data across the network in a way that allows crackers to read it—and they either send your password along, visible to the crackers, or they expect computers to allow access without passwords.

SSH is different; it was designed with security in mind. If anything you do over a network (like the Internet) is at all confidential, you really should find SSH programs and learn how to use them. SSH isn't just for Unix systems! There are SSH programs that let you log in and transfer files between Microsoft Windows machines, between Windows and Unix, and more. A good place to get all the details and recommendations for programs is the book *SSH: The Secure Shell*, by Daniel J. Barrett and Richard Silverman (O'Reilly).

Windows from Other Computers

In the section "Remote Logins," you saw how to open a terminal session across a network. The X Window System lets you ask a remote computer to open any kind of X window (not just a plain terminal) on your local system. This is hard or impossible to do with remote login programs such as *telnet*. It's also insecure over a public network such as the Internet.

The **ssh** program, when you use it together with an *SSH agent* program, can open remote windows securely and fairly easily, and without needing to log into the remote computer first. This is called *X forwarding*.

Please show this section to your system or network administrator and ask for advice. Although SSH is secure, X forwarding can be resource-intensive, and the first-time setup can take some work. (Also, this concept may be new to your administrator, or he may just want to be aware of what you're doing.)

For example, let's say Dr. Nelson has a graphical data-analysis program named **datavis** on the remote *biolab.medu.edu* computer. She needs to run it from her local *fuzzy* computer. She could type a command like the following, and (if the first-time setup has been done) a **datavis** window will open on her local system. The connection will be encrypted for security, so no one else can see her data or anything she does to it:

```
fuzzy$ ssh jdnelson@biolab.medu.edu datavis
```

Figure 6-2 shows how this works when the **xterm** program runs on your local computer versus when **ssh** coordinates access to the remote **datavis** program.

Lynx, a Text-based Web Browser

In a window system, you can choose from lots of graphical web browsers: Netscape, Opera, KDE's Konqueror, the browser in StarOffice, and more. If you have a window system, try the various Unix browsers to find one you like. Those browsers don't work without a window system, though. They also can be slow—especially with flashy, graphics-laden web pages on a slow network.

The Lynx web browser (originally from the University of Kansas, and available on many Unix systems) is different, and has tradeoffs you should know about. It works in terminals (where graphical browsers can't) as well as in terminal windows. Lynx indicates where graphics occur in a page layout; you won't see the graphics, but the bits of text that Lynx uses in their place can clutter the screen. Still, because it doesn't have to download or display those graphics, Lynx is *fast*, especially over a dialup modem or busy network connection. Sites with complex multicolumn layouts can be hard to follow with Lynx; a good rule is to just page through

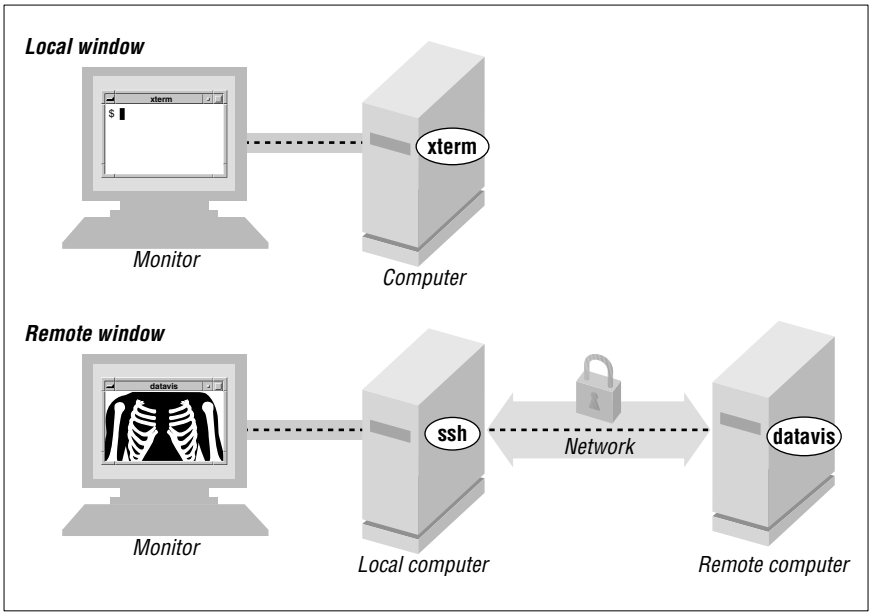

Figure 6-2. Local window, remote window

the screens, looking for the link you want and ignoring the rest. Forms and drop-down lists are a challenge at first—but Lynx always gives you helpful hints for forms and lists, as well as other web page elements, in the third line from the bottom of the screen. With those warts (and others), though, once you get a feel for Lynx you may find yourself choosing to use it—even on a graphical system. Let's take a quick tour.

The Lynx command line syntax is:

lynx "*location***"**

For example, to visit the O'Reilly home page, enter **lynx** "**http://www.oreilly.com**" or simply **lynx** "**www.oreilly.com**". (It's safest to put quotes around the location because many URLs have special characters that the shell might interpret otherwise.) Figure 6-3 shows a part of the home page.

To move around the Web, Lynx uses your keyboard's arrow keys, space bar, and a set of single-letter commands. The third line from the bottom of a Lynx screen gives you a hint of what you might want to do at the moment. In Figure 6-3, for instance, "press space for next page" means you can see the next screenful of this web page by pressing the space bar (at the bottom edge of your keyboard). Lynx doesn't use a scroll bar; instead, use the space bar to go forward in a page, and use the **b**

command to move back to the previous screenful of the same web page. The bottom two lines of the screen remind you of common commands, and the help system (which you get by typing **h**) has the rest.

```
www.oreilly.com -- Welcome to O'Reilly & Associates     (p8 of 14)

Essential SNMP --This guide for network and system administrators
  introduces SNMP, an Internet-standard protocol for managing
  hosts on an IP network. The book's primary focus is on
  network administration. Essential SNMP covers all versions
  through SNMPv3, and it also explores commercial and open source
  packages, including OpenView, SNMPc, and MRTG.  Sample Chapter 2,
  A Closer Look at SNMP, is available online.

  Dreamweaver 4: The Missing Manual is a complete user's guide
  to Macromedia Dreamweaver. This Missing Manual also
  shows how to customize Dreamweaver with libraries, templates,
  shortcuts, and extensions. Sample Chapter 17, Libraries and
  Templates, is available online in PDF format.

-- press space for next page --
  Up and Down keys move.  Right follows a link; Left goes back.
 H)elp O)ptions P)rint G)o Q)uit /=search [delete]=history list
```

Figure 6-3. Lynx display

The links (which you would click on if you were using a graphical web browser) are highlighted. One of those links is the *currently selected link*, which you can think of as the link where your cursor sits. On a monochrome terminal, links are boldfaced and the selected link (in Figure 6-3, that's the first "Essential SNMP") is in reverse video. Emphasized text is also boldfaced on monochrome terminals, but you won't be able to select it as you move through the links on the page. On a color terminal, links are blue, the selected link is red, and emphasized text is pink.

When you first view a screen, the link nearest the top is selected. Figure 6-4 shows what you can do at a selected link. To select a later link (farther down the page), press the down-arrow key. The up-arrow key selects the previous link (farther up the page). Once you've selected a link you want to visit, press the right-arrow key to follow that link; the new page appears. Go back to the previous page by pressing the left-arrow key (from any selected link; it doesn't matter which one).

Although Lynx can't display graphics in a terminal (*no* program can!), it will let you download links that point to graphical files—such as the last link in Figure 6-3, for instance, Then you can use other Unix programs—such as **gimp** or **xv** (for graphics), and **acroread** (for PDF documents)—to view or print those files.

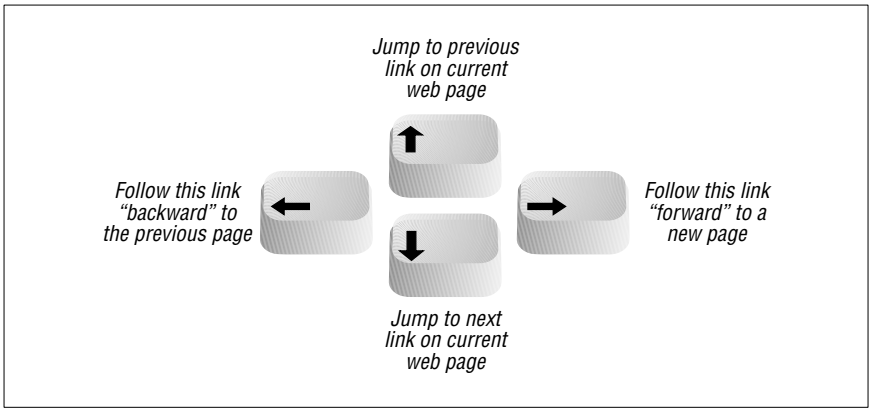

Figure 6-4. Lynx link navigation with the arrow keys

There's much more to Lynx; type H for an overview. Lynx command-line options let you configure almost everything. For a list of options, type **man lynx** (see the section "Documentation" in Chapter 8) or use:

```
$ lynx -help | less
```

Transferring Files

You may need to copy files between computers. For instance, you can put a backup copy of an important file you're editing onto an account at a computer in another building, or another city. Dr. Nelson could put a copy of a data file from her local computer onto a central computer, where her colleagues can access it. Or you might want to download 20 files from an FTP server, but not want to go through the tedious process of clicking on them one-by-one in a web browser window. If you need to do this sort of thing often, your system administrator may be able to set up a networked filesystem connection; then you'll be able to use local programs such as **cp** and **mv**. But Unix systems also have command-line tools for transferring files between computers. These often do it more quickly than working with graphical tools does. We explore them later in this section.

scp and rcp

Your system may have an **scp** (secure copy) or **rcp** (remote copy) program for copying files between two computers. In general, you must have accounts on both computers to use these. The syntax of **scp** and **rcp** are like **cp**, but also let you add the remote hostname to the start of a file or directory pathname. The syntax of each argument is:

hostname:*pathname*

hostname: is needed only for remote files. You can copy from a remote computer to the local computer, from the local computer to a remote computer, or between two remote computers.

The **scp** program is much more secure than **rcp**, so we suggest using **scp** to transfer private files over insecure networks such as the Internet. For privacy, **scp** encrypts the file and your passphrase.

For example, let's copy the files named *report.may* and *report.june* from your home directory on the computer named *giraffe* and put the copies into your working directory (**.**) on the machine you're logged in to now. If you haven't set up the SSH agent that lets you use **scp** without typing your passphrase, **scp** will ask you:

```
$ scp giraffe:report.may giraffe:report.june .
Enter passphrase for RSA key 'jpeek@home':
```

To use wildcards in the remote filenames, put quotation marks ("*name*") around each remote name.* You can use absolute or relative pathnames; if you use relative pathnames, they start from your home directory on the remote system. For example, to copy all files from your *food/lunch* subdirectory on your *giraffe* account into your working directory (**.**) on the local account, enter:

```
$ scp "giraffe:food/lunch/*" .
```

Unlike **cp**, most versions of **scp** and **rcp** don't have an **-i** safety option. If the files you're copying already exist on the destination system (in the previous example, that's your local machine), those files are overwritten.

If your system has **rcp**, your system administrator may not want you to use it for system security reasons. Another program, **ftp**, is more flexible and secure than **rcp** (but much *less* secure than **scp**).

FTP

FTP, file transfer protocol, is a standard way to transfer files between two computers. The Unix **ftp** program does FTP transfers from the command line.† (Your system may have a friendlier version of **ftp** named **ncftp**. Some graphical filesystem browsers can also handle FTP transfers. But we

* Quotes tell the local shell not to interpret special characters, such as wildcards, in the filename. The wildcards are passed, unquoted, to the remote shell, which interprets them *there*.

† Microsoft Windows, and some other operating systems, have a version of **ftp** that you can use from a command prompt. It works just like the Unix version.

cover the standard **ftp** program here.) Both computers must be connected by a network (such as the Internet), but they don't need to run Unix.

To start FTP, identify yourself to the remote computer by giving the username and password for your account on that remote system. Unfortunately, sending your username and password over a public network means that snoopers may see them—and use them to log into your account on that system.

A special kind of FTP, *anonymous FTP*, happens if you log into the remote server with the username *anonymous*. The password is your email address, like *alex@foo.co.uk*. (The password usually isn't required; it's a courtesy to the remote server.) Anonymous FTP lets anyone log into a remote system and download publicly-accessible files to their local systems.

Command-line ftp

To start the standard Unix **ftp** program, provide the remote computer's hostname:

ftp *hostname*

ftp prompts for your username and password on the remote computer. This is something like a remote login (see the section "Remote Logins," earlier in this chapter), but **ftp** doesn't start your usual shell. Instead, **ftp** prints its own prompt and uses a special set of commands for transferring files. Table 6-1 lists the most important **ftp** commands.

Table 6-1. Some ftp commands

Command	Description
put *filename*	Copies the file *filename* from your local computer to the remote computer. If you give a second argument, the remote copy will have that name.
mput *filenames*	Copies the named files (you can use wildcards) from local to remote.
get *filename*	Copies the file *filename* from the remote computer to your local computer. If you give a second argument, the local copy will have that name.
mget *filenames*	Copies the named files (you can use wildcards) from remote to local.

Table 6-1. Some ftp commands (continued)

Command	Description
prompt	A "toggle" command that turns prompting on or off during transfers with the **mget** and **mput** commands. By default, **mget** and **mput** will prompt you "**mget** *filename*?" or "**mput** *filename*?" before transferring each file; you answer *y* or *n* each time. Typing **prompt** once, from an "ftp>" prompt, stops the prompting: all files will be transferred without question until the end of the **ftp** session. Or, if prompting is off, typing **prompt** at an "ftp>" prompt resumes prompting.
cd *pathname*	Changes the working directory on the remote machine to *pathname* (**ftp** typically starts at your home directory on the remote machine).
lcd *pathname*	Changes **ftp**'s working directory on the local machine to *pathname*. (**ftp**'s first local working directory is the same working directory from which you started the program.) Note that the **ftp lcd** command changes only **ftp**'s working directory. After you quit **ftp**, your shell's working directory will not have changed.
dir	Lists the remote directory (like **ls -l**).
binary	Tells **ftp** to copy the following file(s) without translation. This preserves pictures, sound, or other data.
ascii	Transfers plain text files, translating data if needed. For instance, during transfers between a Microsoft Windows system (which adds CTRL-M to the end of each line of text) and a Unix system (which doesn't), an **ascii**-mode transfer removes or adds those characters as needed.
quit	Ends the **ftp** session and takes you back to a shell prompt.

Here's an example. Carol uses **ftp** to copy the file *todo* from her *work* subdirectory on her account on the remote computer *rhino*:

```
$ ls
afile   ch2    somefile
$ ftp rhino
Connected to rhino.zoo.edu.
Name (rhino:carol): csmith
Password:
ftp> cd work
ftp> dir
total 3
-rw-r--r--  1 csmith   mgmt    47 Feb  5  2001 for.ed
-rw-r--r--  1 csmith   mgmt   264 Oct 11 12:18 message
-rw-r--r--  1 csmith   mgmt   724 Nov 20 14:53 todo
ftp> get todo
```

```
ftp> quit
$ ls
afile   ch2    somefile    todo
```

We've explored the most basic **ftp** commands here. Entering **help** at an `ftp>` prompt gives a list of all commands; entering **help** followed by an **ftp** command name gives a one-line summary of that command.

FTP with a web browser

If you need a file from a remote site, and you don't need all the control that you get with the **ftp** program, you can use a web browser to download files using anonymous FTP. To do that, make a URL (location) with this syntax:

ftp://*hostname*/*pathname*

For instance, *ftp://somecorp.za/pub/reports/2001.pdf* specifies the file *2001.pdf* from the directory */pub/reports* on the host *somecorp.za*. In most cases, you can also start with just the first part of the URL—such as *ftp://somecorp.za*—and browse your way through the FTP directory tree to find what you want. If your web browser doesn't prompt you to save a file, use its "Save" menu command.

An even faster way to download a file is with the handy Lynx web browser. Its **-dump** option sends a page to the standard output, where you can redirect it to a file or pipe it to another program (see Chapter 5). For example, to save the report in a file named *report.pdf*, enter:

```
$ lynx -dump "ftp://somecorp.za/pub/reports/2001.pdf" > report.pdf
```

Electronic Mail

You may see a notice that says "You have mail" when you first log in to your system, or later, before a shell prompt. Someone has sent you a message or document by *electronic mail* (email). With email, you can compose a message at your terminal and send it to another user or list of users. You also can read any messages that others may have sent to you.

There are a lot of email programs for Unix. If you'll use email often, we recommend that you start with whatever program other people in your group use.

We start with a brief section on addressing email. Next, you'll see how to send mail from a shell prompt with Berkeley **mail**. Then we introduce sending and reading mail with Pine, a popular menu-driven program that works without a window system. If you'd like to try a graphical program

(which we won't discuss here), many web browsers have an email window. All programs' basic principles are the same though, and they all can send and receive messages from each other.*

Addressing an Email Message

Most addresses have this syntax:

username@hostname

username is the person's username, like *jerry*, and *hostname* is either the name of his computer or a central domain name for his entire organization, like *oreilly.com*. On many Unix systems, if the recipient reads email on the same computer you do, you may omit the *@hostname*. (An easy way to get a copy of a message you send is to add your username to the list of addressees.)

Sending Mail from a Shell Prompt

Most Unix systems have a fairly simple program from Berkeley Unix called **Mail** (with an uppercase "M"), **mailx**, or just **mail**. If you enter just the program name at a shell prompt, you can read your email, but its terse interface isn't very friendly. If you enter the program name, followed by addresses as arguments, you can send an email message. This is handy for sending a quick message from your keyboard. But it's best used with redirection (explained in Chapter 5) to email the output of a program or the contents of a file.

To send mail, give the address of each person you want to send a message to, such as one of the following:

Mail *address1 address2 . . .*

mailx *address1 address2 . . .*

mail *address1 address2 . . .*

* Microsoft Windows users have an unfortunate habit of sending email "attachments" made with a Windows-specific program like Microsoft Word. On Unix systems, you can read these messages using popular word processing programs such as StarOffice, but it can be a pain. You might ask Windows users to send plain text messages, which everyone can read without special software.

It's best to use simple addresses such as *username@hostname* on the command line. More complex addresses—with peoples' names or special characters such as < and >—can cause trouble unless you know how to deal with them.

After you enter **mail** and the addresses, if you're sending a message from the keyboard, in most cases the program (depending on how it's set up) prompts you for the subject of the message. Many versions of the program also accept a subject as a command-line argument after the **-s** option; be sure to put quote marks around the subject! Here are two examples of redirection: first sending the restaurant list you made in an earlier example, then sorting the list before you send it:

```
$ mail -s "My favorite restaurants" jerry@oreilly.com < food
$ sort food | mail -s "My favorite restaurants" jerry@oreilly.com
```

If you've redirected the standard input from a pipe or file, as in these two examples, your message will be delivered. Otherwise, *mail* will wait for you to enter the message body. Type in your message, line by line, pressing RETURN after every line. When you've finished entering text, type CTRL-D (just once!) at the start of a new line. You should get the shell prompt at this point, though it might take a few seconds.

```
$ mail alicja@moxco.chi.il.us
Subject: My Chicago trip
Alicja, I will be able to attend your meeting.
Please send me the agenda. Thanks.

Jerry
^D
$
```

If you change your mind before you type CTRL-D, you can cancel a message (while you're still entering text) with your interrupt character (see the section "Correcting a Command Line" in Chapter 1). The cancelled message may be placed in a file called *dead.letter* in your home directory. To see other commands you can use while sending mail, enter **~?** (tilde question mark) at the start of a line of your message, then press RETURN. To redisplay your message after using **~?**, enter **~p** at the start of a line.

You can't cancel a message after you type CTRL-D (unless you're a system administrator and you're lucky to catch the message in time). So, if

you change your mind about Alicja's meeting, you'll need to send her another message.

Please try the previous examples, substituting your address for the sample addresses shown. Once you've found the correct program name and the email address you can use to send a message to yourself, write them down. You'll probably find this is a very useful way to send yourself little reminder messages, the contents of files, and the outputs of programs:

_____ Name of email program that sends from a shell prompt

_____ My email address

Reading Email with Pine

Pine, from the University of Washington, is a popular program for reading and sending email from a terminal. It works completely from your keyboard; you don't need a mouse. Pine isn't a standard part of all Unix systems; if you don't have it, you can use other email programs. If you read this introduction but don't have Pine, ask your system staff to download and install it. Like most Unix software, Pine is free.

Start Pine by entering its name at a shell prompt. It also accepts options and arguments on its command line; to find out more, enter **pine -h** ("help"). If new email is waiting for you, but you want to experiment with Pine without taking chances, the **-o** (lowercase letter "O") option makes your inbox folder read-only; you won't be able to change the messages in it until you quit Pine and restart without the **-o**. Figure 6-5 shows the starting display, the *main menu*.

The highlighted line, which is the default command, gives a list of your email folders.* You can choose the highlighted command by pressing RETURN, pressing the greater-than sign >, or typing the letter next to it (here, **l**, a lowercase L; you don't need to type the commands in uppercase). But since you probably haven't used Pine before, the only interesting folder is the inbox, which is the folder where your new messages wait for you to read them.

* Recent versions of Pine also let you read Usenet newsgroups. The **L** command takes you to another display where you choose the source of the folders, *then* you see the list of folders from that source. See the section "Usenet News," later in this chapter.

```
PINE 4.33   MAIN MENU                        Folder: INBOX  2 Messages

     ?     HELP              -  Get help using Pine

     C     COMPOSE MESSAGE   -  Compose and send/post a message

     I     MESSAGE INDEX     -  View messages in current folder

     L     FOLDER LIST       -  Select a folder OR news group to view

     A     ADDRESS BOOK      -  Update address book

     S     SETUP             -  Configure Pine Options

     Q     QUIT              -  Leave the Pine program

   Copyright 1989-2001.
   PINE is a trademark of the University of Washington.
                  [Folder "INBOX" opened with 2 messages]

? Help                          P To Files   R RelNotes
O OTHER CMDS  > [ListFldrs]     N NextCmd    K KBlock
```

Figure 6-5. Pine main menu

The bottom of the display in Figure 6-5 shows that there are two messages waiting. Let's go directly to the inbox by pressing I (or by highlighting that line in the menu and pressing RETURN) to read the new mail. Figure 6-6 has the *message index* for Alicja's inbox.

```
PINE 4.33          MESSAGE INDEX          Folder: INBOX  Msg 1 of 2

+                  1 May 22 bigboss@moxco  (529) In your spare time
               N   2 Oct  9 Jerry Peek     (362) My Chicago trip

? Help        <  FldrList P PrevMsg   -  PrevPage D Delete   R Reply
O OTHER CMDS  > [ViewMsg] N NextMsg  Spc NextPage U Undelete F Forward
```

Figure 6-6. Pine message index

The main part of the window is a list of the messages in the folder, one message per line. If a line starts with N, like the second message does, it's

a new message that hasn't been read before. (The first message has been sitting in the inbox for some time now . . .) Next on each line is the *message number*; messages in a folder are numbered 1, 2, and so on. That's followed by the date the message was sent, who sent it, the number of characters in the message (size), and finally the message subject.

At the bottom of the display is Pine's reminder list of commands. When you aren't sure what to do, this is a good place to look. If you don't see what you want here, pressing O (the letter "o"; lowercase is fine) shows you more choices. For more information, ? gives detailed help.

Let's skip this first message and read the next one, number 2. The down-arrow key or the N key moves the highlight bar over that message. As usual, you can get the default action—the one shown in brackets at the bottom of the display (here, `[ViewMsg]`)—by pressing RETURN or >. The message from Jerry will appear.

Just as > took us forward in Pine, the < key generally takes you back to where you came from—in this case, the message index. You can type R to reply to this message, F to forward it (send it on to someone else), D to mark it for deletion, and TAB to go to the next message without deleting this one.

When you mark a message for deletion, it stays in the folder message index, marked with a `D` at the left side of its line, until you quit Pine. Type Q to quit. First Pine asks if you really want to quit. If you've marked messages for deletion, Pine asks if you want to *expunge* ("really delete") them. Answering Y here actually deletes the message.

There's much more to Pine than we can cover here. For instance, it lets you organize mail in multiple folders, print, pipe (output) messages to Unix programs, search for messages, and more. Recent versions of Pine can access mail folders on other computers using IMAP; this lets you use Pine (and other email programs) on many computers, but keep one main set of mail folders on a central computer.

Sending Email with Pine

If you're sending a quick message from a shell prompt, you may want to use the method shown in the section "Sending Mail from a Shell Prompt" earlier in this chapter. For a more interactive way to send email, try Pine. We'll take a quick tour.

If you've already started Pine, you can compose a message from many of its displays by typing C. (Though, as always, not every Pine command is

available at every display.) You can also start from the main menu. Or, at a shell prompt, you can go straight into message composition by typing "**pine** *addr1 addr2*", where each *addr* is an email address like *jerry@oreilly.com*. In that case, after you've sent the mail message, Pine quits and leaves you at another shell prompt.

When you compose a message, Pine puts you in a window called the *composer*. (You'll also go into the composer if you use the Reply or Forward commands while you're reading another mail message.) The composer is a lot like the Pico editor, but the first few lines are special because they're the message *header*—the "To:", "Cc:" (courtesy copy), "Attchmnt:" (attached file), and "Subject:" lines. Figure 6-7 shows an example, already filled in.

```
PINE 4.33   COMPOSE MESSAGE                Folder: <CLOSED>  No Msgs

To      : jpeek@jpeek.com,
          eddie@moxco.chi.il.us
Cc      :
Attchmnt:
Subject : Thoroughly trivial experimentation
----- Message Text -----
In the interest of furthering the educational objectives and
enlightenment of the person whose identity shall be revealed
momentarily, the current electronic communication has been
rendered.  May I obtain a response?

John

P.S.  Yow!!


^G Get Help  ^X Send       ^R Read File  ^K Cut Text    ^O Postpone
^C Cancel    ^J Justify    ^W Where is   ^U UnCut Text  ^T To Spell
```

Figure 6-7. Pine composer

As you fill in the header, the composer works differently than when you're in the message text (body of the message). The list of commands at the bottom of the window is a bit different in those cases, too. For instance, while you edit the header, you can attach a file to the end of the message with the "Attach" command, which is CTRL-J. However, when you edit the body, you can read a file into the place you're currently editing (as opposed to attaching it) with the CTRL-R "Read File" command. But the main difference between editing the body and the header is the way you enter addresses.

If you have more than one address on the same line, separate them with commas (,). Pine will rearrange the addresses so there's just one on each line, as shown in Figure 6-7.

There are several ways to give the composer the addresses where the message should be sent:

- Type the full email address, like *jpeek@jpeek.com*.
- If you're sending email to someone who uses the same computer you do, type their username. Pine will fill in `@hostname` as soon as you move the cursor to the next line.
- Type a nickname from the address book. (See "the section "Pine address book," later in this chapter.)

Move up and down between the header lines with CTRL-N and CTRL-P, or with the up-arrow and down-arrow keys, just as you would in Pico. When you move into the message body (under the "Message Text" line), type any text you want. Paragraphs are usually separated with single blank lines.

If you put a file in your home directory named *.signature* (the name starts with a dot, "."), the composer automatically adds its contents to the end of every message you compose. (Some other Unix email programs work the same way.) You can make this file with a text editor like Pico, or from the Pine setup menu (see the section "Configuring Pine," later in this chapter). It's good Internet etiquette to keep this file short—no more than four or five lines, if possible.

You can use familiar Pico commands such as CTRL-J to justify a paragraph and CTRL-T to check your spelling. When you're done, CTRL-X ("exit") leaves the composer, asking first if you want to send the message you just wrote. Or CTRL-C cancels the message, though you'll be asked if you're sure. If you need to quit, but don't want to send or cancel, the CTRL-O command postpones your message; then, the next time you try to start the composer, Pine asks whether you want to continue the postponed composition.

Pine address book

The Pine *address book* can hold peoples' names and addresses, as well as a *nickname* for each person. When you compose a message, enter peoples' nicknames in the message header; Pine replaces that with the full name and address.*

You can enter information by hand from the main menu by choosing A ("address book"), then adding new entries and editing old ones. Also, as you read email messages that you've received, the T ("take address") command makes new address book entries for that message's addressees.

Figure 6-8 shows the address book entry form. Edit each line as you would in the composer, then use CTRL-X to save the entry. The "Fcc" line gives the name of an optional Pine folder; when you send a message to this address book entry, Pine puts a copy in this folder. (If you leave "Fcc" blank, Pine uses the *sent-mail* folder.) All lines except nickname and address are optional.

```
Nickname  : Jerry
Fullname  : Jerry Peek
Fcc       : authors
Comment   : Writes books about Unix and the Internet
Addresses : jpeek@jpeek.com
```

Figure 6-8. Pine address book entry

Once you've saved that address book entry, if you go into the composer and type the nickname *Jerry*, here's the header you get automatically:

```
To      : Jerry Peek <jpeek@jpeek.com>
Cc      :
Fcc     : authors
Attchmnt:
Subject :
```

Configuring Pine

The Pine main menu (shown in Figure 6-5) has a Setup entry for configuring Pine. We assume that your system staff has configured important options, like your printer command, and we look at a few other settings you might want to change.

---

* Recent versions of Pine also let you store your address book on a central server, in order for you to access it, from whatever other computer you're using at the moment, via IMAP.

After you enter S (the "Setup" command), you can choose what kind of setup you want to do. From the setup screen, you can get to the option configuration area with C (the "Config" command).

The configuration screen has page after page of options. You can page through them with the space bar (to move forward one page), the - key (back one page), the N key (to move forward to the next entry), and the P key (back to the previous entry). If you know the name of an option you want to change, you can search for it with W (the "Whereis" command).

When you highlight an option, the menu of commands at the bottom of the screen will show you what can do with that particular option. A good choice, while you're exploring, is the ? ("Help") command, to find out about the option you've highlighted. There are several kinds of options:

- Options with variable values: names of files, hostnames of computers, and so on. For example, the *personal-name* option sets the name used in the "From:" header field of mail messages you send. The setup entry looks like this:

  ```
  personal-name        = <No Value Set: using "Robert L. Stevenson">
  ```

  "No Value Set" can mean that Pine is using the default from the system-wide settings, as it is here. If this user wants his email to come from "Bob Stevenson," he could use the C ("Change Val") command to set that name.

- Options that set preferences for various parts of Pine. For instance, the *enable-sigdashes* option in the "Composer Preferences" section puts two dashes and a space on the line before your default signature. The option line looks like this:

  ```
  [X]  enable-sigdashes
  ```

  The "X" means that this preference is set, or "on." If you want to turn this option off, use the X ("Set/Unset") command to toggle the setting.

- For a few options, you can choose one of many possible settings. The option appears as a series of lines. For instance, the first few lines of the *saved-msg-name-rule* option look like this:

  ```
  saved-msg-name-rule      =
             Set    Rule Values
             ---  ----------------------
             (*)  by-from
             ( )  by-nick-of-from
  ```

```
( )  by-nick-of-from-then-from
( )  by-fcc-of-from
( )  by-fcc-of-from-then-from
```

The "*" means that the *saved-msg-name-rule* option is currently set to *by-from*. (Messages will be saved to a folder named for the person who sent the message.) If you wanted to choose a different setting—for instance, *by-fcc-of-from*—you'd move the highlight to that line and use the [*] ("Select") command to choose that setting.

These settings are trickier than the others, but the built-in help command [?] explains each choice in detail. Start by highlighting the option name (here, *saved-msg-name-rule*) and reading its help file. Then look through the settings' names, highlight one you might want, and read its help file to see if it's right for you.

When you exit the setup screen with the [E] command, Pine asks you to confirm whether you want to save any option changes you made. Answer [N] if you were just experimenting or aren't sure.

Exercise: sending and reading mail

You can practice sending and reading mail in this exercise:

List logged-in users.	Enter **who**
Choose a user you know, else choose yourself; send a short message to that person using **mail** or your favorite email program.	Enter **mail** *username* or **pine** *username* or . . .
Read the message or messages you got.	Enter **pine** or start your favorite email program; use its "read message" commands.
Reply to one of the messages. (It's okay to reply to a message from yourself.)	Press **R** in **pine** or use your email program's "reply" command. Send the completed reply.
Forward one of the messages. (It's okay to forward a message to yourself.)	Press **F** in **pine** or use your email program's "forward" command. Add a sentence or two of explanation above the forwarded message. Send the completed message.

Usenet News

Usenet, also called "Net News," has thousands of worldwide discussion groups. Each discussion is carried on as a series of messages in its own *newsgroup*. A newsgroup is named for the kind of discussion that happens there. Each message is a lot like an email message. But, instead of being sent to a list of email addresses, a newsgroup message is sent to all

the computers that subscribe to that particular newsgroup—and any user with access to that computer can read and reply to the message.

Because Usenet is a public forum, you'll find a variety of people with a variety of opinions—some impolite, rude, or worse. Although most users are friendly and helpful, a few people seem to cause most of the problems. Until you're accustomed to Usenet, be aware that you may be offended.

To read Usenet groups, you'll need a *newsreader* program, also called a *news client*. Many email programs can read news, too. You can use any newsreader; the principles of all are about the same. Some of the more popular Unix newsreaders are **slrn**, **nn**, and **trn**. We show how to read news with Pine Version 4.33.* If you haven't used Pine before, please read the section "Reading Email with Pine," earlier in this chapter.

If your system's copy of Pine has been set up to read Usenet messages, when you choose the L key ("folder list") from the main menu, you'll get a Collection List screen like Figure 6-9. A *collection* is a group of folders. A collection can be email folders from your local computer, email folders from other computers, or Usenet newsgroup folders. Figure 6-9 shows two collections: *Mail* and *News on news/nntp*. The News collection is selected (highlighted).

```
PINE 4.33   COLLECTION LIST                    Folder: INBOX  No Messages

    Mail
        Local folders in mail/

    News on news/nntp
        News groups on news/nntp




? Help          < Main Menu    P PrevCltn     - PrevPage
O OTHER CMDS    > [View Cltn]  N NextCltn   Spc NextPage   W WhereIs
```

Figure 6-9. Pine collection list screen

* Much older versions of Pine can't show newsgroups. Choose another newsreader or upgrade to the newest Pine.

If your copy of Pine is recent enough to read Usenet, but doesn't seem to do it, check the configuration settings, as described in the section "Configuring Pine," earlier in this chapter. The *collectionList* settings can set up a collection of folders for news. You may also need to set the *nntp-server* hostname to the computer which serves news articles; your system staff should be able to tell you the right hostname.

When you press ENTER or > to view that collection, you'll get a list of newsgroup folders that's probably huge. Usenet has something for everyone! The Pine D command will delete a newsgroup from your list; it won't appear anymore unless you use the A command to add it back. (Pine also has some advanced features, like "zooming" to a list of folders that you've defined. See the Pine help system for details.) Figure 6-10 shows a list of some newsgroups.

```
PINE 4.33   FOLDER LIST                        Folder: INBOX  No Messages

                        News groups on news/nntp
-------------------------------------------------------------------------

alt.3d
alt.activism
alt.activism.d
alt.activism.death-penalty
alt.adoption
alt.alien.visitors
alt.alt
alt.amateur-comp
alt.amazon-women.admirers
alt.amiga.demos
alt.angst
alt.animals.badgers
alt.animals.bears
alt.animals.dolphins
alt.animals.felines


? Help      < ExitSubscb   P PrevFldr     - PrevPage L List Mode
      S [Subscribe] N NextFldr   Spc NextPage            W WhereIs
```

Figure 6-10. Pine newsgroup collection list screen

Newsgroup names are in a hierarchy, with names of the levels separated by dots (.):

- The main hierarchies include *comp* (for discussions about computers); organization, city, regional and national groups (such as *ne* for New England, *uk* for the United Kingdom, and so on); *misc*

("miscellaneous"); and so on. The *alt* ("alternative") hierarchy is for almost anything that doesn't fit in the others.

- All the top levels have subcategories, or second-level categories. For instance, the *alt* category has subcategories *alt.3d*, *alt.activism*, *alt.adoption*, and so on, as you can see in Figure 6-10.
- A second-level category may have third-level categories. For instance, the category *alt.animals* is divided into *alt.animals.badgers*, *alt.animals.bears*, and so on.

When you first start to read Usenet, it's a good idea to spend a couple of hours exploring what's available and what you're interested in—and deleting unwanted newsgroups from your list. The time you spend at first will pay you back later, by letting you go straight to the newsgroups in which you're interested.

People all over the world frequent particular newsgroups. Just as mail folders have email messages, newsgroups have *news articles* (individual messages posted by someone). These messages *expire* after a period of time. (That's part of why a lot of newsgroups appear empty.) Let's look into a newsgroup. Go to the newsgroup *news.announce.newusers*; scroll through the folder list by pressing the space bar, or if in a hurry, use the W ("whereis") command and enter the newsgroup name. Once you've selected the name from the collection list, press ENTER or > to view it. You'll see a list of messages in the group, as in Figure 6-11.

```
PINE 4.33   MESSAGE INDEX   news.announce.newusers  Msg 1 of 4


     1 Jul 15 David Alex La (49,945) FAQ: How to find people's E-mail
     2 Aug 13 *.answers mod (19,102) Introduction to the *.answers ne
     3 Aug 14 Chris Lewis   (32,675) How to become a Usenet site
     4 Aug 15 news.announce  (2,591) Welcome to newsgroups and Usenet


[News group "news.announce.newusers" opened with 4 messages]

? Help          <  FldrList  P  PrevMsg    -  PrevPage  D  Delete
O OTHER CMDS    >  [ViewMsg]  N  NextMsg  Spc  NextPage  U  Undelete
```

Figure 6-11. Pine newsgroup message index screen

Read Usenet messages just as you read email messages; for example, select a message from the message index and press ENTER or > to view it. It stays in the index until it's deleted or expires. Deleting messages you've read or don't want to see makes it easier to find new messages that come in later. To keep a message, save a copy to a Pine mail folder with the S ("save") command, email a copy to other users with the F ("forward") command, or save a copy to a file with the E ("export") command.

Remember that people worldwide will see your message and have your email address. If your message is insulting, long and rambling, includes a lot of the original message unnecessarily, or just makes people unhappy, you're likely to get a lot of email about it. Many newsgroups have periodic FAQ ("frequently asked questions") postings that give more information about the group and answer common questions. We suggest that you not post messages to newsgroups until you've read Usenet for a while, have learned what style is acceptable, and have seen enough of the discussion in a particular group to know whether your question or comment has been discussed recently.

Also, remember that spammers (people who send "junk email" with advertising and worse) will be able to see the email address on your Usenet posting. For that reason, many people set a different email address in the "From:" field when posting Usenet messages. If your Internet provider gives you multiple email addresses, you could choose one just for your Usenet postings. (Readers may want to reply to your message by email, though, so consider using an email address that you do read occasionally. You also can include your "real" address in the body of the article, possibly disguised to fool spammers who search Usenet articles for email addresses.)

If there's a message you want to reply to, the Pine R command starts a reply. After asking whether to include a copy of the original message in your reply, Pine asks you: "Follow-up to news group(s), Reply via email to author or Both?" If you want all who read this newsgroup to see your reply, choose F to follow up; your reply, including your name and email address, is posted for everyone to see. If your message is just for the author—for instance, a question or a comment—replying by email with R is the better choice.

You can post a new message to a newsgroup with the C ("compose") command. If you're viewing a news folder, Pine asks if you want to compose a message to that newsgroup. (If you answer N ("no"), Pine creates a regular email message.)

Here's one more tip: to read expired messages or search through years of archives, web sites like Google Groups (*http://groups.google.com/*) allow this.

Interactive Chat

Need a quick answer from another user without sending an email message and waiting for his reply? Want to have a conversation with your Internet-connected friend in Chile but don't have money for an international phone call? An interactive chat program lets you type text to another user and see her reply moments later. Chatting, or "instant messaging," has become popular recently. Widely known chat programs are available for Unix; as of this writing, those include Jabber and AOL Instant Messenger. Other programs have been available on Unix systems for years. We look at two of these: **talk** and IRC.

talk

The **talk** program is simple to use. Give the username (and, optionally, the hostname) of the person you want to chat with. Then **talk** will try to notify that person as well as show how to use **talk** to complete the connection with you. Both of your terminal windows will be split into two sections, one for the text you type and the other for the text you get from the other person. You can type messages back and forth until one of you uses CTRL-C to break the session.

One advantage of **talk** is its simplicity; if each of you has a terminal window open, either of you can run the program at any time; if the other person is logged in, he is notified that you want to chat and told how to complete the connection. If both people want to use **talk** on the same computer—even if one of them is logged in remotely (see the section "Remote Logins," earlier)—it should work well. Unfortunately, there are several **talk** versions that don't work with each other. So, the first time you try to chat with someone on another host, which might have another **talk** version (or other problems), it can take planning. Use an email message or phone call to alert them that you'll try **talk**ing soon, then experiment to be sure that both of you have compatible **talk** systems. After that, you're all set.

Here's the syntax:

talk *username@hostname*

If the other user is logged onto the same computer as you, omit the *@hostname*. After you run that command, your screen clears with a line of dashes across the middle. The top half shows text you type and informational messages about the connection. The bottom half shows what the other user types.

For example, if your username is *juan*, you're logged onto the computer *sandya.unm.edu*, and you want to talk to the user *ana* at the computer *cielo.cl*, you would type "**talk ana@cielo.cl**". If the connection works, your screen clears and you'll see something like Figure 6-12.

```
[No connection yet]
[Waiting for your party to respond]
[Waiting for your party to respond]
[Connection established]
Hi, Ana!  Need any help with your exam?



------------------------------------------------------------
```

Figure 6-12. A successful talk connection

The message *[Waiting for your party to respond]* means that your **talk** program has found *ana*'s system and is waiting for her to respond. Ana's terminal bell should ring and she should see a message like this in one of her terminal windows:

```
Message from Talk_Daemon@sandya.unm.edu at 18:57 ...
talk: connection requested by juan@sandya.unm.edu.
talk: respond with:  talk juan@sandya.unm.edu
```

If she answers by typing **talk juan@sandya.unm.edu**, the connection should be completed, and her screen should clear and look like Juan's. What she types appears on the top half of her screen and the bottom half of Juan's, and vice versa. It's not always easy to know when the other person has finished typing; one convention is to type *o* (for "over") when you want a response; type *oo* (for "over and out") when you're finished. The

conversation goes on until one person types CTRL-C to actually break the connection.

Unfortunately, because there are several versions of **talk**, and because other things can go wrong, you may see other messages from the **talk** program. One common message is *[Checking for invitation on caller's machine]*, which usually means that you won't be able to connect. If this happens, it's possible that one system has other versions of the **talk** program that will work with the particular system you're trying to connect to—try the **ntalk** program, for instance. It might also be easier to use a more flexible chat system, such as IRC.

IRC

IRC (Internet Relay Chat) is a long-established system for chatting with other users worldwide. IRC is fairly complex, with some rules you need to understand before using it. We give a brief introduction here; for more details, see *http://www.irchelp.org*.

Introducing IRC

Unlike the **talk** program, IRC programs let you talk with multiple users on multiple channels. Channels have names, usually starting with "`#`", such as *#football*. (You might hope that a channel name would tell you what sort of discussions happen there, but you'd often be wrong!) Many channels are shared between multiple servers on an IRC *net*, or network; you connect your IRC program to a nearby server, which spreads your channel to other servers around the net. Some channel names start with "`&`"; these channels are local to their server, and not shared around the net. Finally, you can meet a user from a channel and have a private conversation, a "DCC chat," that doesn't go through servers.

Each user on a channel has a *nick*, or nickname, which is up to 9 characters long. It's a good idea to choose a unique nick. Even when you do, if someone else with the same nick joins a channel before you do, you must choose another nick.

Two kinds of users are in control of each channel. *Ops*, or channel operators, choose which other users can join a channel (by "banning" some users from joining) and which users have to leave (by "kicking off" those users). If a channel is empty, the first user to join it is automatically the channel op. (As you can imagine, this system means that some ops can be arbitrary or unhelpful. If an op treats you badly, though, you can just go join another of the thousands of IRC channels.) *IRC ops*, on the other

hand, are technical people in charge of the servers themselves; they don't get involved with "people issues."

IRC not only lets you chat; it lets you share files with other users. This can be helpful, but it also can be dangerous; see the Warning later in this section.

There are many IRC programs, or "clients," for different operating systems. They all work with each other, though some have more features. The best known Unix program is ircII, which you run by typing **irc**. Another well-liked program, based on ircII, is **bitchx**; get it from *http://www.bitchx.org*. Many programs can be modified by using *scripts* or *bots*; there are thousands of these floating around IRC. But we advise you to use only well-known programs, and to avoid scripts and bots, unless you know that they're safe.

IRC started long before graphical programs were popular. IRC programs use commands that start with a slash (/), such as **/join #football** or **/whois StevieNix**. Some IRC programs have buttons and menus that run commands without typing, but you'll probably find that learning the most common commands is easy—and makes chatting faster, overall, than using a mouse.

IRC can be a wide open security hole if you don't use it carefully. If you type the wrong command or use an insecure program or script, any user can take over your account, delete all of your files, and more. Be careful!

IRC programs can be corrupted; scripts and bots can easily do damage. Even if you think that one is widely known and safe, it can contain a few lines of dangerous "trojan horse" code added by an unscrupulous user. Also, never type a command that another IRC user suggests unless you're sure you know what it does; **/load** and **/dcc get** can be especially dangerous.

Finally, you should know that IRC users can get information about you with the **/whois** *nick* command, where *nick* is your current nick. They'll see your real name unless you set the IRCNAME environment variable to another name (and log into your system again to make the change take effect). This is explained in the section "Customizing Your Account," in Chapter 3. (By the way, use **/whois** with your nick to find out what other people can see about you.)

A sample IRC session

When you type **irc**, your terminal screen splits into two parts. The top part shows what's happening on the server and the channel; the bottom part (a single line) is where you type commands and text. In between the two parts is a status line with the time of day, your nick, and other information. Some terminals can't do what **irc** wants them to; if you get an error message about this, try the command **irc -d** to use "dumb mode" instead.

A good ircII command to start with is **/help**, which provides a list of other commands. The commands **/help intro** and **/help newuser** give introductions. For help with a particular command, give its name—such as **/help server** for help with the **/server** command. When you're done with help, you'll get a "Help?" prompt; you can type another help topic name, or simply press RETURN to leave the help system. Another common command is **/motd**, the "message of the day," which often explains the server's policies.

You can type your nick on the **irc** command line. Your IRC program should have a default server. You can change servers with the **/server** command; you'd do this if your server is full (you get the message "connection timed out," "connection refused," etc.). If your default IRC server is down or busy, you can also give a server hostname on the **irc** command line, after your nick.

In the following examples, we show the text you type (from the bottom line of the screen) in **boldface**, followed by the responses you might see (from the top of the screen) in unbolded text.

```
$ irc sstjohn us.undernet.org
*** Connecting to port 6667 of server us.undernet.org
     ...
*** Closing Link: sstjohn by austin.tx.us.undernet.org (Sorry, your
+connection class is full - try again later or try another server)
*** Connecting to port 6667 of server us.undernet.org
     ...
*** Welcome to the Internet Relay Network sstjohn (from
+Arlington.VA.US.Undernet.Org)
     ...
*** on 1 ca 1(4) ft 10(10)

/motd
*** The message of the day was last changed: 27/7/2001
*** on 1 ca 1(4) ft 10(10)
*** - Arlington.VA.US.Undernet.Org Message of the Day -
*** - 27/7/2001 20:39
     ...
*** -          SERVER POLICIES:
```

```
     ...

/help newuser
*** Help on newuser
     ...
*** Hit any key for more, 'q' to quit ***
     ...
Help? RETURN

/whois sstjohn
*** sstjohn is ~jpeek@kumquat.jpeek.com (Steve St. John)
*** on irc via server *.undernet.org (The Undernet Underworld)
*** sstjohn has been idle 1 minutes
```

Messages from the server start with ***. Long lines are broken and continue on following lines that start with +. After connecting to the server, I used **/whois** with my nick to find what information other users could see about me. The Undernet servers have thousands of channels open, so I started by searching for channels with "help" in their names; you can use wildcards, such as **help**, to do this:

```
/list *help*
*** Channel    Users  Topic
*** #helpmania 2      A yellow light, an open door, hello neighbor,
+there's room for more. English
*** #underneth 14     -= UndernetHelp =- Ask your color free questions
+& wait for it to be answered. (undernethelp@fivemile.org)
*** #mIRCHelp  14     Welcome to Undernet's mIRC Help Channel! Beginners
+welcome :-)
*** #irc_help  48     Welcome to #irc_help. We do not assist in
+questions/channels regarding warez, mp3, porn, fserve, etc.
     ...list goes on and on...

/list *mp3*
     ...list of groups discussing/sharing MP3 files...
```

I want to see what's happening, so I join the biggest help channel: *#irc_help*, which has 48 users now:

```
/join #irc_help
*** sstjohn (jpeek@kumquat.jpeek.com) has joined channel #irc_help
*** Topic for #irc_help: Welcome to #irc_help. We do not assist in
+questions/channels regarding warez, mp3, porn, fserve, etc.
*** Users on #irc_help: sstjohn ChuckieCheese Dodgerl GooberZ
+Kinger MotorMouth @theDRJoker MrBean SweetPea LavaBoy GrandapaJoe
     ...
```

Some names in the list of users, like *@Darkmind*, start with `@`; these users are ops. Let's watch some more of the action. After a couple of users leave

the channel, a new user *MsTiger* joins and asks for help. Each time a user types a line of text that isn't a command, it's sent to everyone else on the channel, preceded by that user's nick, like *<MsTiger>*:

```
*** ChuckieCheese has left channel #irc_help
*** GooberZ has left channel #irc_help
*** HelloWorld (~hw@foo.edu) has joined channel #irc_help
*** MsTiger (~tiger@zz.ro) has joined channel #irc_help
<MsTiger> help me
<MsTiger> please
<Kinger> MsTiger what can we help you with ?
<MsTiger> my channel is not op
<Kinger> LavaBoy tell MsTiger about no opers
<LavaBoy> MsTiger, *shrug*
<GrandapaJoe> MsTiger Sorry, but there are currently NO IRC Operators
+available to help you with your channels. Please be patient and wait
+for an Operator to join.
*** MsTiger has left channel #irc_help
```

The channel has gotten quiet, so I jump in with a question:

```
Hello all. When I joined, I had a problem
    ...
Any suggestions??
*** Thor (dfdddd@194.999.231.00) has joined channel #irc_help
<[Wizard]> Can you help me plz
<LavaBoy> Try typing !help in the channel, [MORTAL].
/leave
*** sstjohn has left channel #irc_help
/quit
$
```

No one had an answer, so I left the channel after a few minutes of waiting. Other channels might be a lot livelier, and might have had someone willing to chat about my question, but I left the **irc** program by typing **/quit**. Then I got another shell prompt.