

*Unlocking the Power of Unix*

*Learning*

# Unix *for* Mac OS X Tiger



O'REILLY®

*Dave Taylor*

---

# Taking Unix Online

A network lets computers communicate with each other, share files, send email, and much more. Unix systems have been networked for more than 25 years, and the Mac OS has had networking as an integral part of the system design from day one. In fact, AppleTalk was the first computer network that let computers connect directly together without needing a server in the middle.

This chapter introduces Unix networking: remotely accessing your Mac from other computers and copying files between computers. It also shows you how the Terminal's "Connect to Server" feature can make common connections a breeze once you've set them up the first time.

## Remote Logins

There may be times when you need to access your Mac, but you can't get to the desk it's sitting on. If you're working on a different computer, you may not have the time or inclination to stop what you're doing, walk over to your Mac, and log in (laziness may not be the only reason for this: perhaps someone else is using your Mac when you need to get on it, or perhaps your Mac is miles away). Mac OS X's file sharing (System Preferences → Sharing → Services) lets you access your files, but there may be times you want to use the computer interactively, perhaps to move files around, search for a particular file, or perform a system maintenance task.

If you enable Remote Login (System Preferences → Sharing → Services), as shown in Figure 8-1, you can access your Mac's Unix shell from any networked computer that can run the Secure Shell, SSH.

The *ssh* client program is included with Mac OS X (access it from within Terminal) and all Unix and Linux systems. And just in case you need to

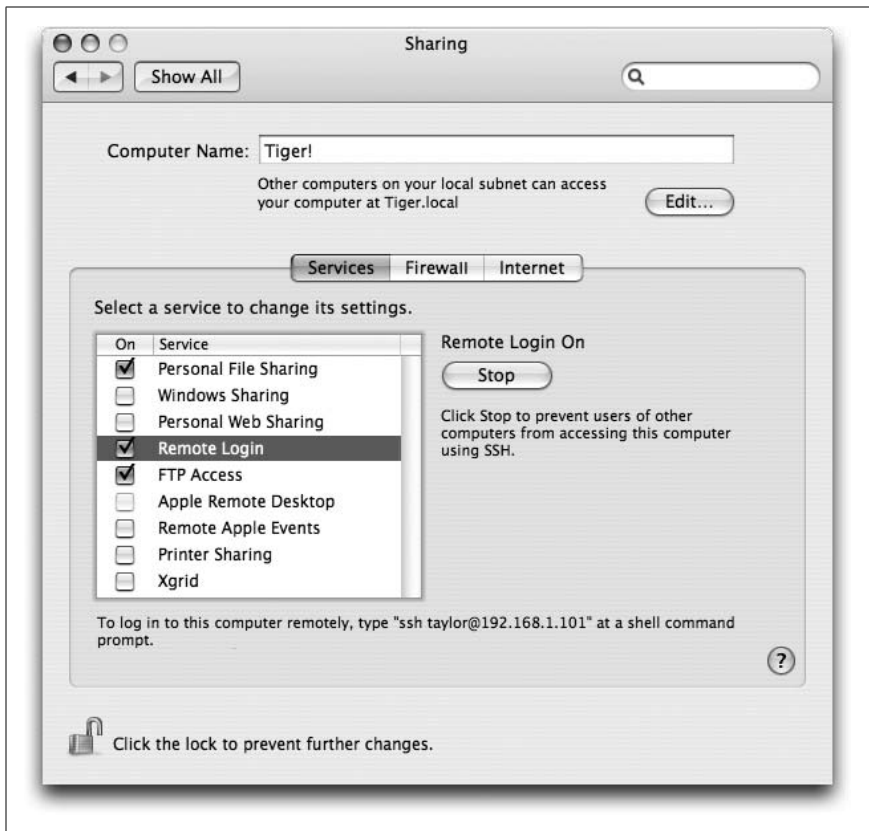


Figure 8-1. Enabling Remote Login in the Sharing preferences panel

access your Mac from a Windows system, there are a number of different *ssh* applications available, including:

- SSH ([www.ssh.com](http://www.ssh.com))
- OpenSSH ([www.openssh.org](http://www.openssh.org))
- PuTTY ([www.chiark.greenend.org.uk/~sgtatham/putty/](http://www.chiark.greenend.org.uk/~sgtatham/putty/))

Figure 8-2 shows how remote login programs such as *ssh* work. In a local login, you interact directly with the shell with the Terminal application. In a remote login, you run a remote-access program (such as SSH) on your local system, and that program lets you interact with a shell program on the remote system. When you enable Remote Login, the Sharing panel displays instructions for logging into your Mac from another computer. This message is shown in Figure 8-1 near the bottom:

To log in to this computer remotely, type “ssh taylor@192.168.1.101” at a shell command prompt.

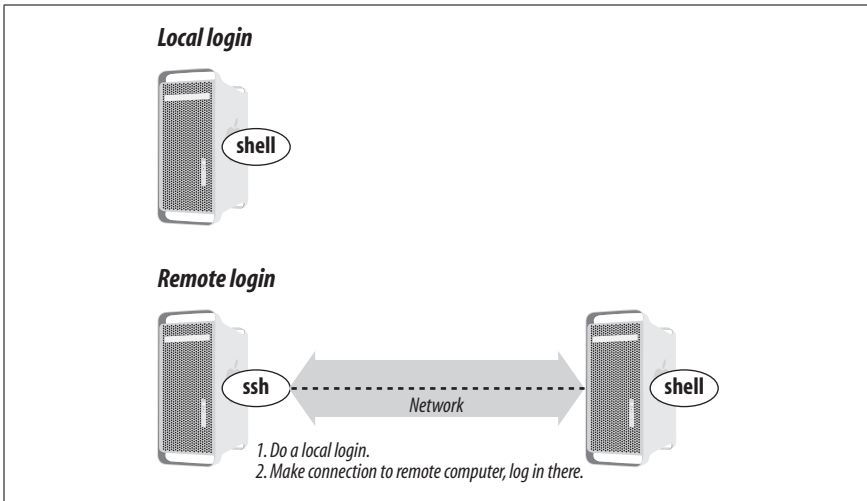
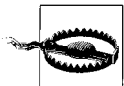


Figure 8-2. Local login, remote login

To log into your Mac from a remote Unix system, use the command displayed in the Sharing panel, as shown in the following sample session. Here, a user on a Red Hat Linux system is connecting to a Mac OS X computer (the first time you connect, you'll be asked to vouch for your Mac's authenticity):

```
Red Hat: taylor $ ssh taylor@192.168.1.101
The authenticity of host '192.168.1.101 (192.168.1.101)' can't be
established.
RSA key fingerprint is 86:f6:96:f9:22:50:ea:4c:02:0c:58:a7:e4:a8:10:67.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.101' (RSA) to the list of known hosts.
taylor@192.168.1.101's password:
Last login: Thu Sep 25 10:27:58 2003
Welcome to Darwin!
$
```



If you have a firewall running, you need to open up a network port to allow remote connections. You can learn more about how to do this by starting with Apple's Tiger Help system. In the Finder, use Command-? to launch Help Viewer, then search for "firewall".

To log into your Mac from a Windows machine using PuTTY, launch the PuTTY application, specify SSH (the default is to use the Telnet protocol described later), and type in your Mac OS X system's IP address, as shown in the Mac's Sharing panel. PuTTY prompts you for your Mac OS X username and password. Figure 8-3 shows a sample PuTTY session.

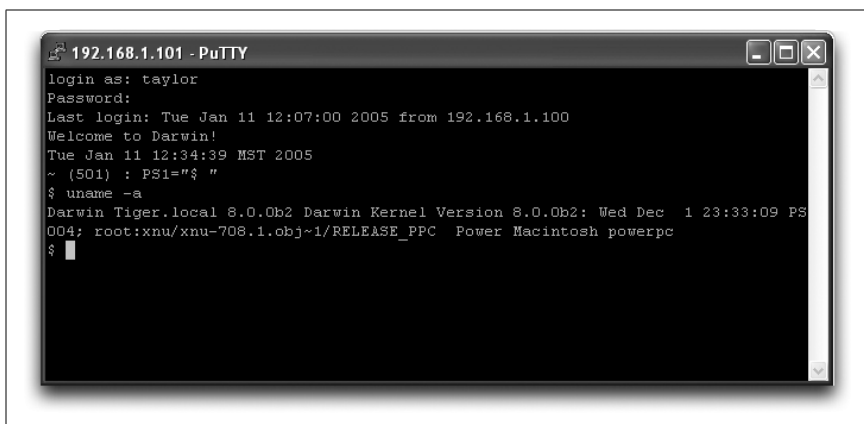



Figure 8-3. Connecting to Mac OS X with PuTTY

For the most part, being connected via *ssh* is identical to using the Terminal application itself. You can even use the *open* command (discussed in Chapter 7) to launch applications on the Macintosh system, which can surprise the heck out of anyone who might be watching the screen! Of course, you won't be able to use the applications if you're remote.



Using Apple Remote Desktop or Virtual Network Computing (VNC) software, you can use Mac applications remotely. You can learn more about Apple Remote Desktop with the Help Viewer (launch Help Viewer with -? in the Finder, then search for "Remote Desktop") or Chicken of the VNC by starting at <http://sourceforge.net/projects/cotvnc/>. If these look like they'll meet your remote access needs, there's a lot more information on remote application access in *Mac OS X Tiger for Unix Geeks* (O'Reilly).

One of the very few differences is that the system records the Internet address of the system from which you're connected remotely, as shown in this *who* output:

```
$ who
taylor console Jan 13 16:56
taylor ttyp1 Jan 13 17:00
taylor ttyp2 Jan 13 17:10 (192.168.1.100)
```

The third entry is a remote connection by a user on a different computer.

## Web and FTP Access

You can also use the Sharing preferences panel to enable your system's web and FTP server. To enable the Apache web server, you'll need to turn on Personal Web Sharing in the Services pane. Other users can access the main home page (located in `/Library/WebServer/Documents`) using `http://address`, where *address* is your machine's IP address or hostname (see the sidebar "Remote Access and the Outside World" if you are using an AirPort Base Station (either Extreme or Express) or other router between your network and the Internet).

### Remote Access and the Outside World

If your Macintosh has an IP address that was assigned by an AirPort Base Station, then it's probable that your machine is inaccessible to the outside world. Because of this, you will be able to connect to your Mac only from machines on your local network. You can allow remote users to connect by using the AirPort Admin Utility, and then follow these steps:

1. Select your Base Station.
2. Click the Configure button in the toolbar.
3. Select the Port Mapping tab.
4. Click the Add button to add a port that you want to map to an IP address on your local network.

For Remote Login via *ssh*, you must map port 22 to your Macintosh; use port 80 for Personal Web Sharing. Other SoHo (Small Office/Home Office) gateways may support this feature as well.

If you use this technique, the IP address shown in the Sharing preferences panel will be incorrect. You should use your AirPort Base Station's WAN address when you connect from a computer outside your local network.

To allow remote users FTP access to your Mac, you'll need to turn on the FTP Access option in the Sharing → Services panel. Once enabled, remote users can use your machine's IP address or hostname to connect to your Mac to download and upload files via FTP.

Later on in this chapter, I'll explain more about how to use the *ftp* program from the command line, but Figure 8-4 gives you a sneak preview of what it's like to connect from a Windows machine to a Mac OS X Tiger system using Firefox's built-in FTP capability. To learn more about Firefox and what it's capable of, visit the Mozilla Project page ([www.mozilla.org/](http://www.mozilla.org/))

products/firefox/central.html), or read *Don't Click on the Blue E: Switching to Firefox*, by Scott Granneman (O'Reilly).

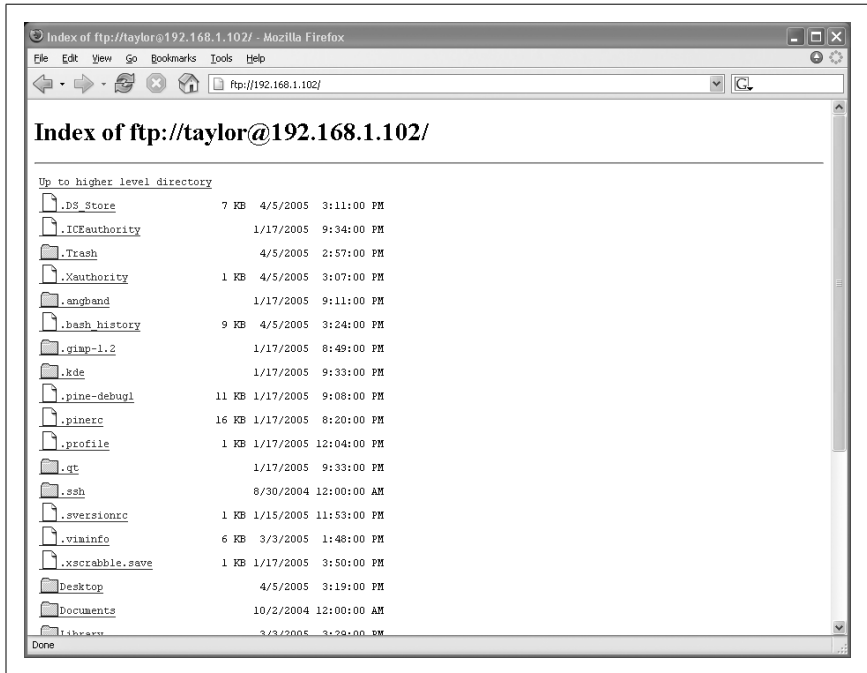
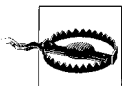


Figure 8-4. Connecting to Mac OS X FTP using Firefox in Windows

## Remote Access to Other Unix Systems

You can also connect to other systems from Mac OS X. To do so, launch the Terminal application, and then start a program that connects to the remote computer. In addition to *ssh*, some typical programs for connecting over a computer network include *telnet*, *rsh* (remote shell), or *rlogin* (remote login). All of these are supported and included with Mac OS X Tiger. In any case, when you log off the remote computer, the remote login program quits and you get another shell prompt from your Mac in the Terminal window.



While you can use *ssh*, *telnet*, *rsh*, or *rlogin* to connect to a remote system, security experts highly discourage you using anything other than *ssh*, because none of the others are encrypted or secure. This means that when you type in your account and password information they're sent "in the clear" through the Internet to the remote system, exposing you to possible "sniffers" who would then be able to log in as if they were you. Better safe than sorry: insist that the remote system support *ssh* and use it exclusively.

The syntax for *ssh* is:

```
ssh remote-user@remote-hostname
```

For example, when Dr. Nelson wants to connect to the remote computer named *biolab.medu.edu*, her first step is to launch the Terminal. Next, she'll need to use the *ssh* program to reach the remote computer. Her session would look something like this:

```
Welcome to Darwin!

$ ssh nelson@biolab.medu.edu
nelson@biolab.medu.edu's password:

biolab$
.
.
.
biolab$ exit
Connection to biolab.medu.edu closed.
$
```

As you can see, the shell prompt from her account on the *biolab* server includes the hostname. This is helpful, because it reminds her when she's logged in remotely, and after exiting the remote system, she'll also know when she's back in her own territory. If you use more than one system but don't have the hostname in your prompt, see "Setting Your Prompt" in Chapter 2 to find out how to add it.

When you're logged on to a remote system, keep in mind that the commands you type take effect on the remote system, not on your local one! For instance, if you use *lpr* to print a file (see the section on printing in Chapter 6), the printer it comes out from won't be the one sitting under your desk, but quite possibly one that's 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, you'll need to provide your remote username on the command line of the remote login program, as shown earlier for *ssh*.

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

Following are four sample *ssh* and *rsh* command lines. The first pair shows how to log into the remote system, *biolab.medu.edu*, when your username is

---

\* In *ssh*, you can run an *agent* program, such as *ssh-agent*, that asks for your passphrase once, then handles authentication every time you run *ssh* or *scp* afterward.

the same on both the local and remote systems. The second pair shows how to log in if your remote username is different (in this case, *jdnelson*); note that the Mac OS X versions of *ssh* and *rsh* may support both syntaxes shown, depending on how the remote host is configured.

```
$ 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 who try to break into computers and snoop on other network users. While the popular media calls these people *hackers*, the correct term to use is *crackers*. (Most hackers are self-respecting programmers who enjoy pushing the envelope of technology.)

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, including your password, across the network in a way that allows even the most inexperienced cracker to read it. Worse, some of these utilities can be configured to allow access without passwords.

SSH is different; it was designed with security in mind. It sends your password (and everything else transmitted or received during your SSH session) in a secure way. For more details on SSH, I'd recommend the book, *SSH, The Secure Shell: The Definitive Guide*, by Daniel J. Barrett and Richard Silverman (O'Reilly).

## 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. Or, Dr. Nelson could copy a 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 don't want to go through the tedious process of clicking on them one by one in a web browser.

If you need to do this sort of thing often, you may be able to set up a networked filesystem connection; then you'll be able to use the Finder or local programs such as *cp* and *mv* to help you move files around on your own

system. But Unix systems also have command-line tools such as *scp* and *rcp* for transferring files between computers. These often work more quickly than most graphical applications, and believe it or not, they're pretty easy to use, as we'll explore in this section.

## scp and rcp

Mac OS X includes both *scp* (secure copy) and *rcp* (remote copy) programs for copying files between two computers. In general, you must have accounts on both computers to use these commands. The syntax of *scp* and *rcp* are similar to *cp*, but they 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 I suggest using *scp* to transfer private files over insecure networks such as the Internet. For privacy, *scp* encrypts the file and your passphrase during the transfer of the data.

For example, let's copy the files *report.may* and *report.june* from your home directory on the computer named *w2.intuitive.com* and put the copies into your working directory (.) on the machine you're presently logged into. If you haven't set up an SSH agent that lets you use *scp* without typing your passphrase, *scp* asks you:

```
$ scp w2.intuitive.com:report.may w2.intuitive.com:report.june .  
Enter passphrase for RSA key 'taylor@mac':
```

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 *w2* account into your working directory (.) on the local account, enter:

```
$ scp "w2.intuitive.com:food/lunch/*" .
```

Unlike *cp*, the Mac OS X 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.

\* 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*.

To be safe, always use *ls* to check what's in the destination directory before you copy files.

Two useful command options for use with *scp* are *-p*, which has the creation and modification dates of the file preserved in the copy, and *-r*, which lets you recursively copy folders and their contents to the remote system. For example, to copy everything in my *Pictures* directory to the *w2* server:

```
$ scp -r ~/Pictures w2.intuitive.com:.  
$
```

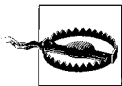
If your system has *rcp*, your system administrator may not want you to use it for system security reasons. Another program, *ftp*, is more flexible and secure than *rcp* (but much less secure than *scp*).

## FTP

The File Transfer Protocol, or FTP, is a standard way to transfer files between two computers. Many users of earlier Mac OS versions are familiar with Fetch ([www.fetchsoftworks.com](http://www.fetchsoftworks.com)), a shareware graphical FTP client that runs on all versions of Mac OS. There are also a number of graphical FTP programs available from the Apple web site (go to “Mac OS X Software” from the Apple menu, and then click on Internet Utilities in the web page that opens). While Fetch offers an easy-to-use interface, it also comes with a price tag, which begs the question: why spend your hard-earned cash on Fetch when you get FTP services for free with Unix?

The Unix *ftp* program does FTP transfers from the command line. But since it's fast, easy, and portable, I'll cover the standard *ftp* program here.

To start FTP, identify yourself to the remote computer by giving the username and password for your account on that remote system.



Sending your username and password over a public network with *ftp* means that snoopers might see them—and then use them to log into your account on that system. Instead, you should use *sftp*, because it uses SSH for an encrypted, secure FTP connection.

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, such as *taylor@intuitive.com*. (The password isn't usually 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.

Here's how that might look:

```
$ ftp ftp.apple.com
Trying 17.254.16.10...
Connected to ftp.apple.com.
220 ProFTPD 1.2.9 Server (Apple Anonymous FTP Server) [ftp01.apple.com]
Name (ftp.apple.com:taylor): anonymous
331 Anonymous login ok, send your complete email address as your password.
Password:taylor@intuitive.com
230 Anonymous access granted, restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
229 Entering Extended Passive Mode (|||49378|)
150 Opening ASCII mode data connection for file list
drwxrwxrwx  3 ftpprod  ftpprod           102 May  7  2003 Apple_Support_Area
drwxrwxr-x  20 ftpprod  ftpprod           680 Aug  28  2003 developer
drwxrwxr-x  37 ftpprod  ftpprod          1258 May  18  2004 emagic
drwxrwxr-x  11 ftpprod  ftpprod           374 Mar  9  2004 filemaker
drwxrwxrwx  10 ftpprod  ftpprod           340 Apr  7  2003 research
226 Transfer complete.
ftp> bye
221 Goodbye.
$
```

While my email address is shown here as the password, in practice, *ftp* doesn't show you what you're typing in response to the password prompt, so you need to guess that you're entering the information properly. Once connected, the *dir* command lists files on the remote system. You can also type *help* at the *ftp>* prompt to get a summary of all the commands available. When you're ready to quit, type *bye*.

## 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 "Remote Logins," earlier in this chapter), but *ftp* doesn't start your usual shell. Instead, *ftp* has its own prompt and uses a special set of commands for transferring files. Table 8-1 lists the most important *ftp* commands.

Table 8-1. Some *ftp* commands

Command	Description
<code>put filename</code>	Copies the file <i>filename</i> from your local computer to the remote computer. If you give a second argument, the remote copy will have that name.
<code>mput filenames</code>	Copies the named files (you can use wildcards) from the local computer to the remote computer.
<code>get filename</code>	Copies the file <i>filename</i> from the remote computer to your local computer. If you give a second argument, the local copy will have that name.
<code>mget filenames</code>	Copies the named files (you can use wildcards) from the remote computer to the local computer.
<code>prompt</code>	A “toggle” command that turns prompting on or off during transfers with the <i>mget</i> and <i>mput</i> commands. By default, <i>mget</i> and <i>mput</i> will prompt you with <i>mget filename?</i> or <i>mput filename?</i> before transferring each file; you answer <i>y</i> or <i>n</i> each time. Typing <i>prompt</i> once, from an <code>ftp&gt;</code> prompt, stops the prompting; all files will be transferred without question until the end of the <i>ftp</i> session. Or, if prompting is off, typing <i>prompt</i> at an <code>ftp&gt;</code> prompt resumes prompting.
<code>hash</code>	Displays progress marks on file uploads and downloads so you can gauge progress. Particularly helpful with large transfers.
<code>cd pathname</code>	Changes the working directory on the remote machine to <i>pathname</i> ( <i>ftp</i> typically starts at your home directory on the remote machine).
<code>lcd pathname</code>	Changes <i>ftp</i> 's working directory on the local machine to <i>pathname</i> . ( <i>ftp</i> 's first local working directory is the same working directory from which you started the program.) Note that <i>ftp</i> 's <code>lcd</code> command changes only <i>ftp</i> 's working directory. After you quit <i>ftp</i> , your shell's working directory will not have changed.
<code>dir</code>	Lists the remote directory (like <code>ls -l</code> ).
<code>binary</code>	Tells <i>ftp</i> to copy the file(s) that follow it without translation. This preserves pictures, sound, or other data.
<code>ascii</code>	Transfers plain-text files, translating data if needed. For instance, during transfers between a Microsoft Windows system (which adds Control-M to the end of each line of text) and a Unix system (which doesn't), an <i>ascii</i> -mode transfer removes or adds those characters as needed.
<code>passive</code>	Toggles the setting of passive mode. This may help <i>ftp</i> to run correctly if you are behind a firewall. If you put the command <code>export FTPMODE=passive</code> in your <i>.profile</i> file, all your FTP sessions will use passive mode.
<code>quit</code> or <code>bye</code>	Ends the <i>ftp</i> session and takes you back to a shell prompt.
<code>!cmd</code>	Gives the specified command to a shell, displays its output, then returns to the <i>ftp</i> program.

Here's an example. Kiana moves into the local directory she wants to use as a starting point (a good idea whether you're uploading or downloading). She then lists the files in her current directory to see what's there, and then uses *ftp* to connect to an FTP server, located at *rhino.zoo.edu*. After using her username and password to log on, Kiana changes directories to the *work* subdirectory, and she then gets the *todo* file and downloads that to her local

machine. After receiving the “Transfer complete” message, Kiana uses the *!ls* command to make sure that the file she transferred is on her local machine, and with the knowledge that the file is there, she quits the FTP session.

```
$ cd uploads
$ ls
afile  ch2  somefile
$ ftp rhino.zoo.edu
Connected to rhino.zoo.edu.
Name (rhino:kiana): ktaylor
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
local: todo remote: todo
227 Entering Passive Mode (17,254,16,11,224,18).
150 Opening BINARY mode data connection for todo (724 bytes)
226 Transfer complete.
724 bytes received in 00:00 (94.06 KB/s)
ftp> !ls
afile  ch2  somefile  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.

## SFTP: FTP to secure sites

If you can only use *ssh* to connect to a remote site, chances are it won’t support regular FTP transactions either due to higher security. Fortunately, Mac OS X also includes a version of *ftp* that’s part of the *ssh* package and works similarly to regular FTP. To run the program, type *sftp* at the command line. Here’s an example:

```
$ cd downloads
$ sftp taylor@intuitive.com
Connecting to intuitive.com...
The authenticity of host 'intuitive.com (128.121.96.233)' can't be
established.
RSA key fingerprint is d0:db:8b:cb:73:c8:37:e4:9a:71:fc:7a:e2:d6:40:81.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'intuitive.com,128.121.96.233' (RSA) to the list
of known hosts.
taylor@intuitive.com's password:
```

```

sftp> cd mybin
sftp> dir -l
drwxr-xr-x  0 24810   100      1024 Jun 26 20:18 .
drwxr-xr-x  0 24810   100     1536 Sep 16 18:59 ..
-rw-r--r--  0 24810   100       140 Jan 17  2003 .library.account.info
-rwxr-xr-x  0 24810   100     3312 Jan 27  2003 addvirtual
-rw-r--r--  0 24810   100       406 Jan 24  2003 trimmailbox.sh
-rwxr-xr-x  0 24810   100     1841 Jan 24  2003 unpacker
-rwxr-xr-x  0 24810   100       946 Jan 22  2003 webspell
sftp> get webspell
webspell                                100% 946    4.7KB/s   00:00
sftp> quit
$ ls -l webspell
-rwxr-xr-x  1 taylor  taylor  946 25 Sep 11:28 webspell

```

The *sftp* program also has a very useful option, which you can specify when you're copying files: *-P* causes the program to preserve date and time information:

```
sftp> get -P webspell
```

Additional helpful commands include *lcd*, *lls*, and *lmkdir* to change your location in the local filesystem, list the files in the current local working directory, and make a new local directory. You can also use the *!* escape to access any Unix command from within *sftp*. Like the *ftp* program, *sftp* also has built-in help, which you can access by typing **help** at the prompt.

## 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, enter a URL (location) with this syntax:

```
ftp://hostname/pathname
```

For instance, *ftp://somecorp.za/pub/reports/2001.pdf* downloads the file *2001.pdf* from the directory */pub/reports* on the host *somecorp.za*. In most cases, you can 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.



If you are using the Safari browser, it will open *ftp:* directories by mounting them in the Finder identically to if you specified the *ftp* URL in the Finder itself, as explained later in this chapter.

## FTP with curl

A faster way to download a file is with the *curl* (copy from URL) command. For example, to save a copy of the report in the current directory, enter:

```
$ curl -O ftp://somecorp.za/pub/reports/2001.pdf
```

Without the *-O* option (that's a capital letter O, not a zero), *curl* dumps the file to standard output (your screen). If you want to read a text file from an Internet server, you can combine *curl* and *less*:

```
$ curl ftp://ftp.oreilly.com/pub/README.ftp | less
```

You can also use *curl* with web pages, but this brings the page up in HTML source view:

```
$ curl http://www.oreilly.com | less
```

One strategy you could use, though it isn't necessarily optimal, is to save HTML pages locally, then open them in Safari:

```
$ curl http://www.oreilly.com > oreilly.home.page.html  
$ open oreilly.home.page.html
```

It turns out that there are better ways to work with HTML pages on the command line, as you'll see in Chapter 10.

## FTP from the Finder

You can also mount remote FTP directories using the Finder, and then access them with standard Unix commands in the Terminal. In the Finder choose Go → Connect to Server... then type in *ftp://* followed by the name of the server that you want to access (such as *ftp.oreilly.com*). Figure 8-5 shows how this appears in Tiger's Finder.

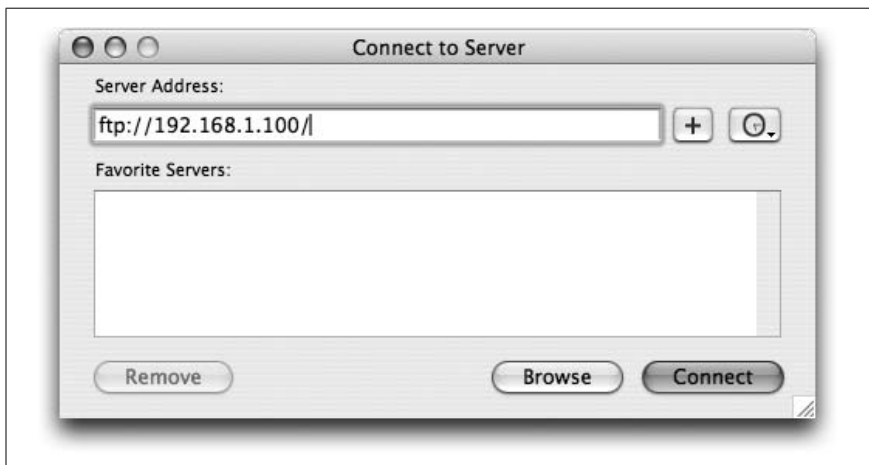


Figure 8-5. Connecting to an FTP server in the Finder

If a password is required, another window pops up, asking you to authenticate with a valid username and password. Enter those correctly and the new FTP disk appears on your Desktop, and is accessible in the */Volumes* directory, as shown here:

```
$ ls -l /Volumes/
total 9
dr-xr-xr-x  1 taylor  unknown   512 Jan 13 20:10 192.168.1.100
drwxrwxr-t 39 root    admin    1326 Jan 11 20:40 Hello
drwxr-xr-x  2 taylor  admin     68 Jan 13 19:18 idisk
lrwxr-xr-x  1 root    admin     1 Jan 13 16:23 shuttle -> /
```

When you're done with the FTP server, you can use the *umount* command to disconnect:

```
$ umount /Volumes/192.168.1.100
```

It's considerably easier than using the *ftp* program!

### Other FTP solutions

One of the pleasures of working with Unix within the Mac OS X environment is that there are a wealth of great Aqua applications. In the world of FTP-based file transfer, the choices are all uniformly excellent, starting with Fetch, NetFinder, Transmit, Cyberduck, Rbrowser, and Anarchie, and encompassing many other possibilities. Either open the Apple menu and select Mac OS X Software, or try VersionTracker ([www.versiontracker.com](http://www.versiontracker.com)), Mac OS X Apps ([www.macosxapps.com](http://www.macosxapps.com)), MacUpdate ([www.macupdate.com](http://www.macupdate.com)), or the shareware archive site, Download.com ([www.download.com](http://www.download.com)).

## Transferring Files with Your iDisk

If you have an account with Apple's popular .Mac system, you'll be interested to learn that you can use Terminal commands to mount your iDisk and then standard Unix commands to copy files to and from the remote device. You can't use *ftp* or *sftp*, because neither of those file transfer protocols are available. Instead, iDisk is based on a different networked filesystem called WebDAV (which stands for the clunky name Web Distributed Authoring and Versioning).

The first step is to connect your iDisk to your Macintosh, which you can do by using the *mount\_webdav* command, specifying both your account on the remote system and the desired mount point. Unlike the iDisk utility, you'll need to create a destination directory before you run the command: I use *~/iDisk* on my system. My account on .Mac is *dave.taylor*, so here's how my invocation looks:

```
$ mkdir ~/iDisk
$ mount_webdav idisk.mac.com/dave.taylor/ ~/iDisk
```

Rather surprisingly, the command-line program immediately pops up an account and password prompt window, as shown in Figure 8-6.



Figure 8-6. Using WebDAV commands, you can mount your .Mac disk from the command line, but you will be prompted for the correct credentials beforehand

Enter your .Mac account information, and the disk is mounted:

```
$ ls -l ~/iDisk
total 57
-rwxrwxrwx  1 unknown  unknown  10532  9 Jun  2004 About your iDisk
drwxrwxrwx  1 unknown  unknown   2048 14 Sep  2004 Backup/
drwxrwxrwx  1 unknown  unknown   2048 10 Jul  2000 Documents/
drwxrwxrwx  1 unknown  unknown   2048 14 May  2003 Library/
drwxrwxrwx  1 unknown  unknown   2048  4 Jan  2000 Movies/
drwxrwxrwx  1 unknown  unknown   2048  8 Jan  2001 Music/
drwxrwxrwx  1 unknown  unknown   2048  4 Jan  2000 Pictures/
drwxrwxrwx  1 unknown  unknown   2048  4 Jan  2000 Public/
drwxrwxrwx  1 unknown  unknown   2048 23 Oct  2003 Sites/
drwxrwxrwx  1 unknown  unknown   2048 20 Nov  2003 Software/
```

To copy files to or from the network-based iDisk, simply use *cp*. I can easily upload a backup copy of a PDF document I've been writing:

```
$ cp Business-Blogging.pdf iDisk/Documents
$ ls -l iDisk/Documents
total 1353
-rwxrwxrwx  1 unknown  unknown  692661 13 Jan 19:26 Business-Blogging.pdf*
```

When you're done working with your iDisk, simply *umount* it:

```
$ umount ~/iDisk
```

Working with a remote disk couldn't be easier!

## Easy Shortcuts with Connect to Server

The Terminal application has a very helpful feature that can make connecting to remote systems via *telnet*, *ssh*, *ftp*, or *sftp* a breeze, once it's set up. Connect to Server is available in the File menu and is shown in Figure 8-7.



Figure 8-7. Connect to Server offers simple shortcuts

To add a service, click on the + icon on the left side of the window. More commonly, you'll add servers, which you can do by clicking on the + icon on the right side of the window. It produces a window that asks for the host-name or host IP address, which is easily entered, as shown in Figure 8-8.

Once added in one area, the new server is available for all services, so to connect to Apple's anonymous FTP archive site, choose *ftp*, then the new server name, and then enter *ftp* into the User box, as shown in Figure 8-9.

Finally, the connection to Apple's server is a breeze: specify the server, specify the user, and click Connect. The results are shown in Figure 8-10.



Figure 8-8. Adding a new server to Connect to Server



Figure 8-9. apple.com



Figure 8-10. Instant connection to Apple's *ftp* server

## Practice

You can practice your *ftp* skills by connecting to the public FTP archive *ftp.apple.com*. Log in as *ftp* with your email address as the password, then look around. Try downloading a research paper or document. If you have an account on a remote system, try using *rcp* and *scp* to copy files back and forth.