Mastering™ Red Hat Linux 7.1

Arman Danesh

Chapter 22: Connecting Linux to the Internet
Connecting Linux to the Internet

- What Is PPP?
- Required Hardware and Software
- Graphical PPP Connections
- Manual PPP Connections
- Automating Manual Internet Connections
The subject of connecting a Linux system (or any computer system, for that matter) to the Internet is a complex one and a subject that can require a thorough knowledge of that system’s network environment.

Still, for many users, this process is simple enough that both X Windows and Linux work on the first attempt.

This chapter starts with a quick look at PPP and its role in the Internet world. It reviews the hardware and software requirements for connecting to the Internet and then walks you through the mechanics of manually making a PPP connection. Finally, this chapter wraps up by explaining how to automate these connections.

What Is PPP?

Most Internet users are probably familiar with the acronym PPP, simply because the type of account they have with their Internet service provider (ISP) is a PPP account. Many users, though, really don’t understand what PPP is all about.

PPP stands for Point-to-Point Protocol and is designed to provide a method by which TCP/IP is extended across an analog modem connection. In this way, when you are connected to the Internet using PPP, you become part of your ISP’s network, you are an actual host on the Internet, and you have an IP address.

Traditionally, dial-up Internet connections were done using terminal software and Unix shell accounts on central servers. In this environment, the terminal software on the client system merely acted as a display for the server, and only the server had an IP address on the Internet. This contrasts sharply with today’s PPP connections, which bring the Internet right up to your modem.

The great flexibility of Internet connection technologies allows for a wide variety of PPP connection types. You can have PPP connections with fixed or dynamic IP addresses. Connections can use special authentication protocols such as PAP (Password Authentication Protocol) or CHAP (Challenge Handshake Authentication Protocol). They can even use standard clear-text-based prompt-and-response mechanisms. Connections can be made manually or can be made automatically as needed.

This chapter discusses the most common Internet connection scenario: connecting by modem to an ISP that offers PPP connections with dynamically assigned IP addresses.
NOTE

“PPP” and “ISP connections” are used interchangeably in this chapter. However, PPP connections are not limited to connections through ISPs. In fact, PPP is also a common way to connect to corporate or educational networks.

Required Hardware and Software

In order to make PPP work properly, some preparation is needed. Hardware and software needs to be in place and configured before you can get PPP to work properly. Three key components need to be considered:

- A modem needs to be installed, configured, and working.
- PPP support needs to be compiled into the Linux kernel.
- PPP software needs to be installed.

The Modem

Because PPP is designed for dial-up connections, a modem is an essential piece of the PPP puzzle.

Chapter 18, “Using Peripherals,” discusses modems and how to install them, test them, and get them working. Refer to that chapter to get your modem up and running and to make a test connection to be sure everything is in order.

To configure PPP, you need to know the speed of your modem connection and which device it uses in Linux (probably /dev/modem or one of /dev/ttyS0 through /dev/ttyS3).

PPP in the Kernel

The Linux kernel is designed to be highly flexible. It can be compiled to include (or exclude) support for numerous technologies ranging from serial mice right up to networking facilities such as PPP.

In order to make a PPP connection with Linux, it is necessary for the kernel to include PPP support. You can check to see if support is there by watching the
messages that scroll by while the operating system is booting. If you see a series of lines like these:

PPP generic driver version 2.4.0
PPP Deflate Compression module registered
PPP BSD Compression module registered

then PPP is compiled into your kernel. If you find that the messages scroll by too quickly, you can use the `dmesg` command to see the part of the start-up messages that should include the PPP messages:

```
$ dmesg | less
```

**NOTE**

If you find that you need to recompile your kernel to include PPP support, refer to Chapter 20, “Recompiling the Linux Kernel,” which discusses the steps involved in recompiling your kernel.

### Installing PPP Software

Red Hat Linux 7.1 installs PPP software with a complete or default installation.

Two programs are used to establish a PPP connection: `/usr/sbin/pppd` and `/usr/sbin/chat`. In Red Hat Linux 7.1, these are part of the `ppp-2.4.0-2` package, and you can see if these are installed by using the `rpm` command:

```
$ rpm -q ppp
ppp-2.4.0-2
```

If you find that you are lacking either `pppd` or `chat`, you need to install a new set of PPP software before continuing. Mount the Red Hat CD-ROM at a suitable location (such as `/mnt/cdrom`) and then install the package `ppp-2.4.0-2.i386.rpm`:

```
$ rpm -i /mnt/cdrom/RedHat/RPMS/ppp-2.4.0-2.i386.rpm
```

Alternatively, you can download the latest sources for PPP from the `pppd` home page at the Samba download mirror site at ftp://ftp.samba.org/pub/ppp/. The current version of PPP is 2.4.1 and the filename is `ppp-2.4.1.tar.gz`.

You need to expand the archive in a suitable location such as `/tmp` with the command:

```
$ tar xzvf ppp-2.4.1.tar.gz
```
and then read through the README.linux file carefully. Installing a new PPP package involves not only compiling the software but also upgrading your Linux kernel source files and recompiling the Linux kernel to match the version of the PPP software being installed.

Describing the details of this process would take up a whole chapter, so it is best left to the documentation.

TIP

Comprehensive sources for downloading Linux software can be found on numerous Web sites, including those for your distribution, as well as the Tucows Web site at http://www.tucows.com. If you prefer rpm packages, a comprehensive source of these packages is located at http://www.rpmfind.net.

Graphical PPP Connections

There are two major tools available to help you make a PPP connection, designed to help you get on the Internet with your Linux computer: RP3 for GNOME and KPPP for KDE. RP3 on Red Hat Linux 7.1 keeps the human interface as simple as possible. KPPP allows you to configure every part of your Internet connection.

The following sections assume a basic Internet connection where you need few special settings. However, some ISPs do require additional special settings. Both RP3 and KPPP can accommodate special settings as well as diagnostic checks. More information on these optional settings is available later in the sections on pppd and minicom.

RP3

RP3 is the Red Hat graphical PPP management tool. If you have an ISP that can handle Linux connections, you can be set up and connected to the Internet in three easy steps. You can make more sophisticated configuration changes later; you can even set up a terminal to troubleshoot any connection problems that you might have. But RP3 is easy to use. First, you configure it as the root user, and then any regular user on your system can connect to the Internet.
Configuring RP3

By default, access is configured right on the desktop. To start RP3, double-click the Dialup Configuration icon on your desktop or simply run the /usr/bin/rp3-config command from a terminal in the X Window.

Enter the root password if prompted. If this is the first time you’re using RP3, then the Add New Internet Connection Wizard starts. Click Next to continue. If you do not yet have a modem configured, you will see a Select Modem dialog box. Plug your modem into a telephone line. Click Next to continue. As you can see in Figure 22.1, RP3 then checks every device that could be connected to a modem.

TIP

If you do not see the Add New Internet Connection Wizard, but you see an Internet Connections dialog box, click the Add button. Otherwise, you may not have the RP3 package installed.

FIGURE 22.1:
Checking for modems
When RP3 finds a candidate device, it checks automatically for a dial tone. After it runs through each device file, it gives you a result like that shown in Figure 22.2.

![Figure 22.2: RP3 finds a modem.](image)

If it doesn’t find a modem, the next window you’ll see is shown in Figure 22.3. This screen allows you to edit the various settings shown to help Linux find and use your modem. Even if Linux found your modem, you can still access this window by selecting the Modify This Modem Manually… option shown back in Figure 22.2.

**NOTE**
The baud rate is not identical to the speed of your modem. It should typically be four times the speed of your modem, up to 115,200 baud. However, you can set a lower speed to reduce data loss from outside factors such as noise.

Once you have the modem configured, click Next. The next step is to set up the basics of your Internet connection, as shown in Figure 22.4:

- **Account Name** is the name of your ISP.
- **Prefix** is the number dialed on your telephone for outside access. You can leave this blank if no prefix is required.
• Area/Country Code is the numbers required to access the local telephone exchange of your ISP’s access number. If you can reach the ISP through a local telephone call, depending on the rules set by your local telephone company, you may want to leave this entry blank as well.
• Phone Number is the local access number for your ISP.

**Figure 22.3:**
Manual modem configuration

**Figure 22.4:**
Entering ISP access information
Click Next, enter the username and password that you use to log on to your ISP, and then click Next again. In the Other Options window, select Normal ISP or AT&T Global Network Services, and click Next. When you click Finish in the next window, you’re ready to get connected.

**Using RP3**

Once a PPP connection is configured, any regular user on your system can run it. From the GNOME Main Menu button, choose Programs ➢ Internet ➢ RH PPP Dialer. The Choose window opens, as shown in Figure 22.5. Select the name you assigned to the ISP, click OK, and Linux connects to the Internet through your ISP.

**KPPP**

The other major Linux GUI way to access the Internet is through KPPP, the KDE PPP dialer. If you’re in the KDE desktop, click the KDE Main Menu button, then choose Internet ➢ Internet Dialer. In another desktop such as GNOME, open up a command-line window and enter the `/usr/bin/kppp` command. This brings up the KPPP window, shown in Figure 22.6.
Click Setup in this window. When you see the KPPP Configuration window, choose the Accounts tab if necessary, and then click New. This opens the Create A New Account… window, which is customized primarily for European ISPs. Unless you use an ISP in Europe or New Zealand, click Dialog Setup. This brings up the New Account window shown in Figure 22.7, where you can set up a connection to your ISP. In most cases, you need to set up only the items shown in Figure 22.7.

**Figure 22.7:** Setting up an ISP

If you have trouble with your connection, come back to this window. You can configure a number of options under the other tabs shown in Figure 22.7:

- **Dial**  Besides the phone number, you can configure the type of password authentication as well as the programs to execute at various points during your connection.

- **IP**  If your ISP assigns you a static IP address, enter it here.

- **Gateway**  If your ISP doesn’t automatically set up a gateway on its network to the Internet for you, you can enter the associated gateway IP address here.

- **DNS**  If your ISP doesn’t automatically set up its DNS servers for your connection, you can enter the associated DNS IP addresses here.
Login Script  If your ISP requires the use of a login script, you can enter the associated commands here.

Execute  This allows you to set up programs or commands to run at different stages of the connection process.

Accounting  If you have to pay for local telephone access on a per-minute basis, you can enter the associated accounting rules here.

Once everything you need is configured, click OK to return to the KPPP Configuration window shown in Figure 22.8.

NOTE  U.S. readers need to understand that in many other countries, local calls are charged by the minute, which creates different requirements for Internet access.

If you have trouble with your modem setup, come back to this window. A number of helpful tools are available under the various tabs shown:

Accounts  Besides showing the name of your ISP account, this tab lets you get to the log files, which often contain useful messages to help you identify whatever problems may arise.

Device  Sets up the hardware parameters associated with your modem.

Modem  Allows you to test or monitor the modem in various ways.
Misc  Determines the behavior of the PPP daemon when you connect, disconnect, or shut down the X Window.

If you need to enable access by regular users, you can set up suid permissions, which allow all regular users to run either program, without giving everyone full access. When you change the permissions on the /usr/sbin/kppp file as follows

```
# chmod u+s /usr/sbin/kppp
```

all regular users can then run the associated command, /usr/sbin/kppp, to access this utility.

---

**Manual PPP Connections**

The RP3 and KPPP utilities are enough for most users. But if you have problems or want to understand the inner workings of PPP connections in Linux, it is important to understand how the PPP daemon (pppd) and other connection utilities work.

In Linux, the PPP connection is made and maintained by pppd. But pppd assumes that a connection has already been made between your modem and your ISP’s modem, that all the necessary logging in has taken place, and that the ISP’s system is also trying to establish a PPP connection on top of the same modem connection.

This sounds like a lot of conditions, but it really isn’t that bad. On almost every ISP, the connection process goes like this:

1. The modem connection is made.
2. The login process occurs.
3. Once the user is authenticated, the ISP begins to try to make a PPP connection by sending the IP address of the client to the client system.

The pppd program requires steps 1 and 2 to have occurred before step 3 gets under way.

Given this process, it is necessary to make a modem connection to the ISP before attempting to run pppd. Once the connection is made, the PPP connection can be established with pppd.
Making a PPP Connection

Fortunately, the `pppd` software is designed to bring things together and work with the `chat` program to handle the entire connection process. Before attempting to connect, you need to gather some information:

- Are you assigned an IP address each time you connect, or do you have a permanently assigned IP address? Because most ISPs work on the basis of assigning a dynamic IP address with each connection, you will also work on this basis.

- How do you log in to your ISP? One option uses plain-text prompt-and-response logins; another uses a special authentication protocol called PAP. A few use another authentication protocol known as CHAP. If you attempt to dial into your ISP with `minicom` or other terminal software and are presented with a login prompt of some sort, then you are logging in with plain-text prompts. If you see random characters after connecting, or no characters at all, then you are probably using PAP or CHAP authentication.

**NOTE**

All the examples in this section assume that you are working as the root user on your machine. Because the process of connecting via PPP requires manipulation of interfaces and creation and deletion of network routes, the configuration programs involved need to be run by the root user.

**TIP**

Information on permanent IP addresses as well as authentication protocols can also help you configure RP3 or KPPP. Talk to your ISP if you aren’t sure about or don’t already have this information.

Connecting with Plain-Text Prompts

Let’s start with plain-text prompts because they are a little easier to work with.

Because `pppd` by itself can take control of your modem device but cannot perform the actual dialing or logging in, you need a way to issue commands to the modem and provide the necessary login information. This is achieved with the `chat` program. The `chat` program allows you to create a conversational exchange.

For instance, the normal dialing process on a modem is to run a terminal editor and, in an empty terminal window, type a dial command such as `ATDT1234567`. 
The response to this is usually a connect string such as CONNECT 115200, to which the user doesn’t respond.

**NOTE** If you expect a response to an ATDT command, connect your modem. Substitute the number of an ISP or other server that can handle terminal connections for 1234567.

This exchange can be turned into a simple chat script:

```
"" ATDT1234567 CONNECT ""
```

This script is made up of two *expect-send pairs*: "" ATDT1234567 and CONNECT "". An expect-send pair contains two pieces of information separated by a space. In the first pair, chat is told to expect nothing (the first pair of double quotes) and in response to send back the string ATDT1234567. In other words, as soon as the script starts processing, the ATDT command dials the selected number. Next, the second pair is processed, and chat is told to expect the string CONNECT and in response to do nothing (the second pair of double quotes). If this were the complete script, chat would finish at this point and exit.

Of course, your chat script needs to be a bit longer. In order to complete your chat script, you need to find out exactly what your login session looks like. You can do this by logging in with regular terminal software such as minicom.

Login prompts at most ISPs generally take the form

Username:
Password:

or

Login:
Password:

or even

Login:
Password:

Consider the first case as the example in this chapter. If you find that your login prompts are different, substitute accordingly as you work through the section.

**NOTE** Instead of Login: and Password:, some ISPs are set up to send your computer
login: or surname: and ssn: to accommodate different operating systems.
So, what is the next expect-send pair? After connecting, you are presented with a Username: prompt, in response to which you provide a username (let’s say your username is testuser). The expect-send pair for this interaction would be Username: testuser.

Once a username is provided, you get a Password: prompt, to which you provide a password (let’s assume this is testpassword). This produces the expect-send pair Password: testpassword.

For many ISPs this is sufficient, and PPP starts on the ISP’s system after the correct password is entered. On a smaller number of systems, the ISP’s computers present you with a command prompt at which you have to type a command to start PPP. In this case, you need to create an additional pair.

For this example, the complete chat script looks like this:

```
" ATDT1234567 CONNECT " Username: testuser Password: testpassword
```

To use this with the chat program (which is normally in /usr/sbin), you simply provide the script as an argument to chat:

```
/usr/sbin/chat " ATDT1234567 CONNECT " Username: testuser Password: testpassword
```

Note, however, that you don’t want to type this at the command line and press Enter. Without being integrated with pppd and provided access to the modem through that program, chat attempts to chat through the shell. You can use this to test your script, though. Simply type the command at the command prompt. When you see ATDT1234567, type CONNECT and then type Username: You should see testuser in response. Follow this by typing Password: and you should see testpassword in response.

**NOTE** If you need to work out problems with your chat script, try adding the -v flag to your chat command. This sends the output of the script to the system log, where you can analyze the results to find problems.

Once you have a working script, you need to integrate it with pppd. You are going to use a handful of pppd options to do this. (The pppd application takes many additional options, which can be found in the pppd man page.) The options you will use are the following:

- **connect** This option is employed to specify a program or command used to establish a connection on the serial line being utilized. In this case, you use the connect option to specify the chat program and its script.
noipdefault  The default behavior of pppd is to determine the IP address of the local machine on the basis of its hostname. But if the ISP assigns a dynamic IP address, which is more common, then the noipdefault option is used to tell pppd to get the IP address from the remote machine to which it is connecting.

defaultroute  This option tells pppd to add a default route to the system’s routing table, using the remote system (your ISP) as the default gateway. The entry is removed when the connection is broken.

The structure of the pppd command is

$ pppd device speed action chatcommands pppdoptions

If you are connecting with the device /dev/modem and the modem has a top compressed speed of 115200bps (in this case, a 56Kbps modem), then you use the following command to connect with the chat script:

$ pppd /dev/modem 115200 connect '/usr/sbin/chat "" ATDT1234567 CONNECT "" Username: testuser Password: testpassword' noipdefault defaultroute

When this command is issued, the modem should dial and, after connecting, it will authenticate and establish a PPP connection. If this process is successful, two things should happen.

First, issuing the command /sbin/ifconfig with no flags or arguments should return a list of interfaces including a PPP interface similar to this one:

```
ppp0      Link encap:Point-to-Point Protocol
           inet addr:194.209.60.101  P-t-P:194.209.60.97 Mask:255.255.255.255
           UP POINTOPOINT RUNNING  MTU:1500  Metric:1
           RX packets:10 errors:0 dropped:0 overruns:0
           TX packets:11 errors:0 dropped:0 overruns:0
```

Second, entries should be added to the routing table to create a default route through the remote machine (check for this using the /sbin/route command with no flags or arguments):

```
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref    Use Iface
du1.paus.ch  *         255.255.255.255 UH 0    0 0   0 ppp0
default      du1.paus.ch 0  UG 0    0 0   0 ppp0
```

In this example, du1.paus.ch is the remote machine in the PPP connection and is serving as the default gateway.
Connecting with PAP Authentication

Connecting with PAP authentication follows the same basic principles as the text-prompt example you just worked through, except that the method of providing the username and password differs.

The first major difference is that you don’t provide the username and password on the command line as part of a PAP script. Instead, you create entries in a special secrets file that is used during the PAP authentication process. In Red Hat Linux 7.1, this file is /etc/ppp/pap-secrets. By default, the permissions on this file make it readable and writable only by the root user. Other users do not have read access to this file.

The format of entries in the secrets file is normally

username servername password

But since you connected to a server through a telephone modem, the second entry is not required. For instance, if you use the test username and password from the previous section, the entry might be

    testuser * testpassword

The * indicates that this password can be used for connections on any interface.

Once your username and password are in the file, you need to write a new chat script. For most ISPs, as soon as the modem connection is established, PAP authentication can begin. Therefore, your chat script gets simpler:

```
" ATDT12345678 CONNECT "
```

This script dials the ISP and makes sure that a connect message is received before ending the chat program and proceeding on to authentication.

Finally, you need to introduce one more option for pppd: user. The user option indicates which PAP user from the pap-secrets file is to be authenticated. The end result is the following pppd command:

```
$ pppd /dev/modem 57600 connect '/usr/sbin/chat " ATDT1234567 CONNECT' noipdefault defaultroute user testuser
```

As with the earlier text-login example, you can check that everything is in order using ifconfig and route.
Once You Have a Connection

Once you have an Internet connection, you need to make sure that you can fully connect to the Internet. To do this, you need to verify that your DNS services are properly configured to point to your ISP’s nameserver.

This is done by editing two files: `/etc/host.conf` and `/etc/resolv.conf`. Both of these files are discussed more fully in Chapter 28, “Configuring Red Hat Linux 7.1 for an Ethernet Network.” This section covers just the necessary basics of the files so that you can quickly get online.

The file `/etc/host.conf` should contain the following two lines:

```plaintext
order hosts,bind
multi on
```

Next, `/etc/resolv.conf` should contain at least two lines:

```plaintext
search
nameserver 100.100.100.100
```

In your case, replace the IP address `100.100.100.100` with the IP address of your ISP’s DNS server, also known as a nameserver. Your ISP can provide this information. If your ISP provides you with more than one nameserver, create a separate line for each one as follows:

```plaintext
search
nameserver 100.100.100.100
nameserver 200.200.200.200
```

Once this is done, you should be able to resolve names and thus access the Internet. You should also be able to run Internet software such as Netscape and FTP. These programs are discussed in later chapters, such as Chapter 23, “Using the World Wide Web.”

Hanging Up

When you have finished using the Internet, you should hang up to reduce phone costs and online charges. To do this, you need to kill the `pppd` process.

You can find out the process ID of the `pppd` program by using the `ps` command as the root user:

```plaintext
$ ps x | grep pppd
```
The entry for `pppd` should look something like this:

```
1316  ttyS0 S   0:00 /usr/sbin/pppd /dev/modem 115200 connect
       /usr/sbin/chat "" AT
```

The first number is the process ID. You can kill this process with the `kill` command, in this case:

```
$ kill 1316
```

When `pppd` is killed, the modem should hang up.

---

**Automating Manual Internet Connections**

While it is great to be able to establish PPP connections to the Internet, if you have to type the long `pppd` and `chat` commands each time you connect, the practicality of using Linux to connect to the Internet will be limited.

To improve the situation, you can create two scripts: one for dialing and one for hanging up. These two scripts are going to be called `dial` and `hangup` (logical choices, yes?). You will want to place these files in a directory on your path such as `/usr/local/bin`.

Because you haven’t learned about scripting yet, these two scripts are presented here with a brief discussion so that you can get right down to using them. These scripts can be created in any text editor. It is important to make sure that lines that appear as one line here stay as one line in your files.

For the `dial` script, let’s assume that the PAP example earlier in the chapter is the way you want to connect to the Internet.

Once you have created the scripts, you will want to make them executable with the `chmod` command:

```
$ chmod 700 dial hangup
```

The permission 700 makes the script readable, writable, and executable by the root user and no one else. If you want to make the script usable by others in the same group as yours, substitute 750 for 700. If you want to make the script usable by anyone who accesses your Linux system, substitute 755 for 700.
You don’t have to create these scripts from scratch. If you have the ppp rpm package installed, preconfigured scripts are available in Red Hat Linux 7.1 in the /usr/share/doc/ppp-2.4.0/scripts directory. For example, you can set up the ppp-on and ppp-off scripts like the dial and hangup scripts described below.

**The dial Script**

The dial script should look something like this:

```
#!/bin/sh
/usr/sbin/pppd /dev/modem 115200 connect \
'/usr/sbin/chat "" ATDT1234567 CONNECT' \
noipdefault defaultroute user testuser
```

The first line tells Linux to process the script through the Bourne Shell, which is at /bin/sh. The other lines were covered earlier. Make sure to substitute the number of your ISP for 1234567. If necessary, substitute your modem device file for /dev/modem and speed for 115200.

Notice that the pppd command is split across three lines. It is still one command. The backslashes at the ends of the first and second lines of this command indicate that the line breaks shouldn’t be treated as line breaks and that the command continues on the next line of the file. This is done to make the script more readable.

**The hangup Script**

The hangup script should look like this:

```
#!/bin/sh
kill `cat /var/run/ppp0.pid`
```

As with the dial script, you start by specifying that the script should be processed with the Bourne Shell. Next, you kill the process in a slightly different way than you learned before.

You are relying on the fact that the pppd process writes its process ID to a file that usually sits in /var/run on most modern Linux systems. The filename consists of the device name followed by a .pid extension. If you have only one modem and one PPP connection active at a time, you can assume that the device is ppp0 and type /var/run/ppp0.pid directly into your script.
The cat command simply displays the contents of the ppp0.pid file on the standard output. You put the cat command in back quotes to pass the result of the cat command (which is the process ID of the pppd process) to the kill command.

NOTE
You can find the back quote under the tilde (~) in the upper-left corner of a standard U.S. keyboard.

But what happens if you have multiple PPP interfaces and want to be able to specify which one to hang up? A quick modification of the hangup script makes this possible:

```bash
#!/bin/sh
kill `cat /var/run/$1.pid`
```

Here, you have replaced ppp0 with $1. The $1 indicates that the value of the first argument to the script should be placed in this location. Now you can pass the interface name to the script as an argument. The command

```
$ hangup ppp1
```

hangs up the modem that is being used for the ppp1 interface. Remember, the first modem is on ppp0, so you may need to run the `hangup ppp0` command instead.

**Looking Ahead**

In this chapter, you took a big step. You brought your Linux system from the isolation of a lone computer on a desktop and connected it to the world with a PPP connection.

In Chapter 23, “Using the World Wide Web,” you will take advantage of this new capability and learn how to use Netscape, the premier Web browser and Internet client for the Linux platform.

Following that, in Chapter 24, “Reading E-Mail,” you will look at the rich variety of e-mail software available for Linux so that you can choose the one that best supports your style of using e-mail.