Networking and the Internet

Motivation
The major predecessor to Linux, UNIX, was one of the first operating systems to provide access to widely distributed local networks as well as the large Internet network that spans the globe. Today, millions of users and programs share information on these networks for a myriad of reasons, from distributing large computational tasks to exchanging a good recipe for lasagna. Linux has inherited this expectation that the network is a fundamental part of any computer system. To make the best use of these network resources, you should understand the utilities that manage the exchange of information. This chapter describes the most useful network utilities and provides an overview of the worldwide network known as the Internet.

Prerequisites
In order to understand this chapter, you should have already read Chapter 1, “What Is Linux?,” and Chapter 3, “GNU Utilities for Nonprogrammers.” It also helps if you have access to a Linux system so that you can try out the various utilities that I discuss.

Objectives
In this chapter, I’ll show you how to find out what’s on the network, how to talk to other users, how to copy files across a network, and how to execute processes on other computers on the network.

Presentation
This chapter begins with an overview of network concepts and terminology, and then describes the GNU network utilities. We finish up with an examination of the history and uses of the Internet.
9.1 Introduction

A network is an interconnected system of cooperating computers. Through a network, you can share resources with other users via an ever-increasing number of network applications, such as web browsers and electronic mail messaging systems.

There has been a huge explosion of network use since 1990. For example, the client-server paradigm described in Chapter 1, “What Is Linux?,” has been adopted by many of the major computer corporations, and relies heavily on the operating system’s network capabilities to distribute the workload between the server and its clients.

In order to prepare yourself for the advent of widespread networking, it’s important to know the following items:

• common network terminology
• how networks are built
• how to talk to other people on the network
• how to use other computers on the network

This chapter covers all of these issues and more. All of the utilities covered in this chapter can be found in most distributions, but might not have been selected at install time. Check your distribution media or your distribution’s application installation tool to find missing applications. For more detailed information about Linux networking, I highly recommend [Stevens, 1998], [Anderson, 1995], and the networking section of [Nemeth, 2002].

9.2 Building a Network

One of the best ways to understand how modern networks work is to look at how they evolved. Imagine that two people in an office want to hook their computers together so that they can share data. The easiest way to do this is to connect a cable between their serial ports. This is the simplest form of local area network (LAN), and requires virtually no special software or hardware. When one computer wants to send information to the other, it simply sends it out of its serial port (Figure 9–1).
9.2.1 **Ethernets**

To make things a little more interesting, let’s assume that another person wants to tie into the other two guys’ existing network. With three computers in the network, we need an addressing scheme so that the computers can be differentiated. We would also like to keep the number of connections down to a minimum. The most common implementation of this kind of LAN is called an Ethernet®. Ethernet is a hardware standard defining cabling, signaling, and behaviors that allow data to pass across a length of wire. The format of this data is defined by network protocols that we’ll look at a bit later. The Ethernet standard was originally developed by Xerox Corporation and works like this:

- Each computer contains an Ethernet card, which is a special piece of hardware that has a unique Ethernet address.
- Every computer’s Ethernet card is connected to the same single piece of wire.
- When a computer wishes to send a message to another computer with a particular Ethernet address, it broadcasts the message onto the Ethernet together with Ethernet header and trailer information that contain the Ethernet destination address. Only the Ethernet card whose address matches the destination address accepts the message.
- An attempt by two computers to broadcast to the Ethernet at the same time is known as a collision. When a collision occurs, they both wait a random period of time and then try again.

Figure 9–2 is a diagram of an Ethernet.

Ethernet networks can transmit data in the order of tens or hundreds megabytes per second.


9.2.2 Bridges

Let’s assume that the Ethernet in the office works so well that the people in the office next door build themselves an Ethernet too. How does one computer on one Ethernet talk to another computer on another Ethernet? One solution might be to connect a special bit of hardware called a bridge between the networks (Figure 9–3). A bridge passes an Ethernet message between the different segments (wires) of the network as if both segments were a single Ethernet network cable. A bridge is used when you need to extend a network past the allowed length of a single section of wire (limited by signal degradation over a distance).

![Figure 9–3 A bridge.](image)

9.2.3 Routers

The use of bridges facilitates the construction of small serially linked sections of Ethernet, but it’s a pretty inefficient way to link together large numbers of networks. For example, assume that a corporation has four LANs that it wishes to interconnect in an efficient way. Stringing them all together with bridges would cause data to pass across the “middle” sections to get to the ends when hosts on those middle sections have no interest in the data. To pass data directly from the originating network to the destination network, a router can be used. A router is a device that hooks together two or more networks and automatically routes incoming messages to the correct network (Figure 9–4).

9.2.4 Gateways

The final stage in network evolution occurs when many corporations wish to connect their local area networks together into a single, large wide area network (WAN). To do this, several high-capacity routers called gateways are placed throughout the country, and each corporation ties its LAN into the nearest gateway (Figure 9–5).

9.3 Internetworking

In order for a collection of LANs and WANs to be able route information amongst themselves, they must agree upon a networkwide addressing and routing scheme. This large-scale interconnection of different networks is known as internetworking. Any group of two or more networks
connected together across administrative boundaries may properly be called “an internet.” However, the largest and best-known such network has become known as “the Internet.” Universities, large corporations, government offices, and military sites all have computers that are part of the Internet, which are generally linked together by high-speed data links. The largest of these computer systems are joined together to form what is known as the backbone of the Internet. Other smaller establishments link their LANs to the backbone via gateways.
9.3.1 Packet Switching

Today’s digital computer networks are packet-switched networks. When one node on the network sends a message to another node, the message is split up into small packets, each of which can be routed independently (switched) through the network.

These packets contain special information that allows them to be recombined at the destination. They also contain information for routing purposes, including the address of the source and destination nodes. The combined set of protocols is called the Transmission Control Protocol and Internet Protocol (TCP/IP) protocol suite. Linux interprocess communication (IPC) uses TCP/IP to allow Linux processes on different machines to talk to each other.

9.3.2 IP Addresses

Hosts on the Internet, as well as many private internets, also use TCP/IP to send data. While it is most popularly implemented on Ethernet networks, TCP/IP can also be used on other types of media. This makes it useful for connecting different types of networks, because not all computers are connected by Ethernet. For example, some LANs may use the IBM Token Ring system. The IP addressing system therefore uses a hardware-independent labeling scheme; the bridges, routers, and gateways transmit messages based purely on their destination IP address. The IP address is mapped to a physical hardware address only when the message reaches the destination host’s LAN. Thus the computer sending the message does not need to understand hardware-specific information of the computer where the message is to be sent.

The IP addressing mechanism works the same whether or not you actually connect your computers to the Internet. When it sets up a LAN that is to be part of the Internet, an organization must get a unique address range assigned to its computers, a process we will see later.

9.3.2.1 IP (IPv4) Addresses

The most common version of the Internet Protocol (IP) in use today is still version 4. In it, an IP address is a 32-bit value that is written as 4 dot-separated numbers, each number representing 8 of the 32 bits of the address. Because each part represents an 8-bit value, the maximum value it may have is 255. Here’s how his form of the IP address looks:

192.127.63.141

However, due to the explosive growth of the Internet, the seemingly endless supply of 32-bit addresses is quickly being used up. The day will come when this version of IP will no longer allow enough Internet addresses to satisfy the demand. Even when it does, local networks may continue to use IPv4 internally, as will the examples in this chapter after the next section.

9.3.2.2 IPv6 Addresses

In the early 1990s, it became clear that a new generation of IP that allowed for many more addresses would be necessary. Work began to define IPng (IP next generation), and a formal proposal for version 6 of the Internet Protocol was released in 1995.

IPv6 specifies 128-bit addresses. Although the two protocols use addresses of different lengths, both protocols can be used on the same network. This is necessary because the Internet is far too large to coordinate a “cut-over” to a new protocol at any moment in time. A smooth
transition to a new addressing scheme requires the ability to evolve to it gradually rather than to require that we all wake up one day using the new protocol.

IP packets (of both versions) specify a version in the first 4 bits of the packet. Therefore, a computer that “speaks” IPv6 can still recognize and handle an IPv4 packet (if it is configured for both protocols). This allows the two protocols to coexist on the same network. The older machines can be upgraded to IPv6 as implementations become available or as the system administrators have the opportunity to upgrade, without requiring it all to happen simultaneously.

IPv6 addresses are expressed in hexadecimal format (rather than the decimal values used in IPv4), requiring two hex digits for each 8 bits, and are delimited every 16 bits by a colon (rather than a period). An IPv6 address looks like this:

```
C07F:3F8D:F11B:5810:014D:2208:BFFD:1B3D
```

In practice, many IP addresses have 8-bit or even 16-bit portions that are zero, and IPv6 also allows for dropping leading zero values as well as eliminating contiguous 16-bit values of zero. So you can actually wind up with much shorter addresses!

In addition to the addressing changes, IPv6 also provides improvements in routing and automatic configuration. While IPv6 is not currently in wide use, vendors are implementing and testing the new protocol. Over the next few years, IPv6 will be deployed across the Internet. If all goes well, people will not even notice. For more information on IPv6, visit the web site at:

http://www.ipv6.org

### 9.3.3 Naming

These numeric addresses are not very convenient for humans to use to access remote computers. Humans are much more used to naming things (people, pets, and cars). So we have taken to naming our computers as well.

When a hostname is assigned to a particular computer, a correlation can be established between its name and its numeric IP address. This way, a user can type the computer’s name to reference it, and the software can translate this name to an IP address automatically.

The mapping of IP addresses to local host names is kept by the LAN’s system administrator in a file called “/etc/hosts”. To show you what this looks like, here’s a small section of the file from UT Dallas:

```
129.110.41.1  manmax03
129.110.42.1  csservr2
129.110.43.2  ncube01
129.110.43.128 vanguard
129.110.43.129 jupiter
129.110.66.8  neocortex
129.110.102.10 corvette
```
9.3.4 Routing

The Internet Protocol performs two kinds of routing: static and dynamic. Static routing information is kept in the file “/etc/route” and is of the form: “You may get to the destination DEST via the gateway GATE with X hops.” When a router has to forward a message, it can use the information in this file to determine the best route. Dynamic routing information is shared between hosts via the “/etc/routed” or “/etc/gated” daemons.¹ These programs constantly update their local routing tables based on information gleaned from network traffic, and periodically share their information with other neighboring daemons.

9.3.5 Security

It has long been known that the only way to keep any computer secure is to put it in a locked room and not connect it to a network. For most applications, however, this is not practical. The network is not only one of the most useful additions to computing, but also one of the most dangerous. The network provides a path for data to enter and leave the system, but makes no judgment about the use of the data. Therefore, it is up to the users or managers of the system to make sure “not just anyone” can gain access to the system or the data being transferred to or from the system.

9.3.5.1 User Authentication

Authentication of a user is the process of establishing that the user is who he or she claims to be. The most common user authentication mechanism is logging in with a username and password.

When you access a remote machine across a network, you generally must re-authenticate in order to gain access. Another method may be to have a set of systems allow access from any of the other systems in the group by assuming you had to authenticate yourself to (log in on) the first system.

Several of the Linux networking utilities that I discuss later in this chapter allow a user with accounts on several machines to execute a command on one of these machines from another. For example, I have an account on both the “csserv2” and “vanguard” machines at UT Dallas. To execute the date command on the vanguard machine from the csserv2 machine I can use the rsh utility (discussed later in this chapter) as follows:

```bash
$ rsh vanguard date  ...execute date on vanguard.
```

The interesting thing about rsh and a few other utilities is that they are able to obtain a shell on the remote host without requiring a password. They can do this because of a Linux facility called machine equivalence. If you create a file called “.rhosts” in your home directory that contains a list of host names, then any user with the same username as your own may log into your account from these hosts without supplying a password. Both my “csservr2” and “vanguard” home directories contain a file “.rhosts” that includes the following lines:

```plaintext
csservr2.utdallas.edu
vanguard.utdallas.edu
```

¹ A daemon is a fancy term for a constantly running background process that is normally started when the system is booted.
We must use the “official” hostname in the “.rhosts” file which includes the Internet domain name (discussed later in this chapter).

This allows me to execute remote commands from either computer without any hassle. Linux also allows a system administrator to list globally equivalent machines in the file “/etc/hosts.equiv”. Global equivalence means that any user on the listed machines can log into the local host without a password. For example, if the “vanguard” “/etc/hosts.equiv” file contained the lines:

csservr2.utdallas.edu
vanguard.utdallas.edu

then any user on “csservr2” could log into the “vanguard” or execute a remote command on it without a password. Global equivalence should be used with great care (if ever).

9.3.5.2 Data Encryption
Even when a user can provide authentication information to a remote system, another problem is posed by a third-party eavesdropping on the network connection and gaining access to the username and password the user provides to login on the remote system. Most login information, and certainly command and input data, is sent in packets across the network. This is often referred to as sending the data “in the clear” or as “clear text.”

Depending on the type and extent of the network in question, this may or may not be an issue. But consider the wireless network in your local coffee shop, where your e-mail client connects to your ISP’s e-mail server and passes your username and password in clear text to check your mail. Anyone on that network could conceivably copy that data and log in as you.

For this reason, many of the common network data-transfer commands now also come in a secure version that encrypts all data being sent to or received from a remote host. This is accomplished through the use of the Open Secure Shell (OpenSSH). OpenSSH is based on the Open Secure Socket Layer (OpenSSL), originally developed by Netscape Communications, to provide secure web connections so that sensitive information like credit card numbers could be sent across the Internet without fear of copying. Now the code that started the electronic commerce revolution can be used to access your own data on a remote host to keep it from prying eyes. For more information on OpenSSH and OpenSSL, visit their web sites:

http://www.openssh.org
http://www.openssl.org

9.3.6 Ports and Common Services
When one network host talks to another, it does so via a set of numbered ports. Every host supports some standard ports for common uses and allows application programs to create other ports for transient communication. The file “/etc/services” contains a list of the standard ports. Here’s a snippet from the UT Dallas file:

echo 7/tcp
discard 9/tcp sink null
The description of the telnet utility later in this chapter provides some examples where I connected to some of these standard ports.

9.3.7 Network Programming
The Linux interprocess communication allows you to communicate with other programs at a known IP address and port. This facility is described in Chapter 12, “Systems Programming.”

9.4 Identifying Network Users
Linux networking is all about moving around the network and talking to other people. Therefore, a basic thing to learn is how to find out who’s on a particular host. There are several utilities that do this, each with its own strengths:

- **users**, which lists all of the users on your local host
- **who**, which is like users except that it gives you more information
- **w**, which is like who except that it gives you *even more* information
- **hostname**, which displays your local host’s name
- **finger**, which gives information about specific users

The next few subsections describe each of these utilities in turn.

9.4.1 Listing Users: users
The **users** utility (Figure 9–6) simply lists the current users of your local system.

<table>
<thead>
<tr>
<th>Utility: users</th>
</tr>
</thead>
</table>

users displays a simple, terse list of the users on your local host.

Figure 9–6 Description of the users command.
Here’s an example of users in action:

$ users
...display users on the local host.
glass posey
$ 

9.4.2 More User Listings: who and w

The who utility (Figure 9–7) supplies a little more information than the users utility.

Utility: who [ whoFile ] [ am i ]

By default, who displays a list of every user on your local host.

Figure 9–7 Description of the users command.

Here’s an example of who:

$ who
...list all users currently on local host.
posey pts/0 May 15 16:31 (blackfoot.utdall)
glass pts/2 May 17 17:00 (bridge05.utdalla)
$ 

The w utility (Figure 9–8) is just as easy to use.

Utility: w { userId }*

w displays a list that describes what each specified user is doing. In other words, it’s almost the same as who.

Figure 9–8 Description of the w command.

Here’s an example:

$ w
...obtain more detailed information than who.
22:25:35 up 11 days 3 users, load average: 0.08, 0.03, 0.01
USER TTY LOGIN@ IDLE JCPU PCPU WHAT
posey pts/0 22:19 2days 1 csh
glass pts/2 17:48 1 13 1 w
$ w glass
...examine just myself.
22:25:48 up 11 days 3 users, load average: 0.08, 0.03, 0.01
USER TTY LOGIN@ IDLE JCPU PCPU WHAT
glass pts/2 17:48 1 13 1 w
$ 

9.4.3 Your Own Host Name: hostname

To find out the name of your local host, use hostname (Figure 9–9).
Here’s an example:

```
$ hostname
...display my host's name.
bluenote
$ _
```

**9.4.4 Personal Data: finger**

Once you’ve obtained a list of the people on your system, it’s handy to be able to learn a little bit more about them. The **finger** utility allows you to do this (Figure 9–10).

```
$ finger
...finger everyone on the system.
Login  Name       Tty  Idle  Login Time  Where
posey  John Posey  pts/0  2d  Fri 16:31  console
```

**Utility: hostname [ hostName ]**

When used with no parameters, **hostname** displays the name of your local host. A super-user may change this name by supplying the new host name as an argument. For more information about this file, see Chapter 14, “System Administration.”

**Figure 9–9** Description of the **hostname** command.

**Utility: finger {userId }**

**finger** displays information about a list of users that is gleaned from several sources:

- The user’s home directory, startup shell, and full name are read from the password file “/etc/passwd”.
- If the user supplies a file called “.plan” in his/her home directory, its contents are displayed as the user’s “plan”.
- If the user supplies a file called “.project” in his/her home directory, its contents are displayed as the user’s “project”.

If no usernames are listed, **finger** displays information about every user that is currently logged on. You may finger a user on a remote host by using the “@” protocol, in which case the remote host’s finger daemon is used to reply to the local finger’s request.

**Figure 9–10** Description of the **finger** command.

I recommend that you create your own “.plan” and “.project” files in your home directory so that people can “finger” you back. Have fun!

In the following example, I fingered everyone on the system and then fingered myself:

```
$ finger
...finger everyone on the system.
Login  Name       Tty  Idle  Login Time  Where
posey  John Posey  pts/0  2d  Fri 16:31  console
```
9.5 Communicating with Network Users

There are several utilities that allow you to communicate with a user:

- **write**, which allows you to send individual lines to a user, one at a time
- **talk**, which allows you to have an interactive split-screen two-way conversation
- **wall**, which allows you to send a message to everyone on the local host
- **mail**, which allows you to send mail messages

The **mail** utility was described in Chapter 3, “GNU Utilities for Nonprogrammers,” and supports the full standard Internet addressing scheme. The rest of these utilities are described in this section, together with a simple utility called **mesg** that allows you to shield yourself from other people's messages.

9.5.1 Shielding Yourself from Communication: **mesg**

The **write**, **talk**, and **wall** utilities communicate with other users by writing directly to their terminals. You may disable the ability of other users to write to your terminal by using the **mesg** utility (Figure 9–11).

```
Utility: mesg [ n | y ]

mesg allows you to prevent other users from writing to your terminal. It works by modifying the write permission of your tty device. The n and y arguments disable and enable writes, respectively. If no arguments are supplied, your current status is displayed.
```

Figure 9–11 Description of the **mesg** command.
In the following example, `mesg` prevented me from receiving a `write` message:

```bash
$ mesg n  ...protect terminal.
$ write glass ...try to write to myself.
write: You have write permission turned off
$ _
```

**9.5.2 Sending a Line at a Time: write**

The `write` command (Figure 9–12) is a simple utility that allows you to send one line at a time to a named user.

```bash
Message from tim@csservr2 on ttyp2 at 18:04
hi Graham -o- ...from tim.
$ write tim ...initiate a reply.
hi Tim -o- ...from me.
don't forget the movie later -oo- ...from tim.
OK -oo- ...from me.
^D ...end of my input.
$ _
```

**Figure 9–12 Description of the write command.**

In the following example, I received a `write` message from my friend Tim and then initiated my own `write` command to respond. We used the `-o-` (over) and `-oo-` (over and out) conventions for synchronization:

```bash
$ Message from tim@csservr2 on ttyp2 at 18:04
hi Graham -o- ...from tim.
$ write tim ...initiate a reply.
hi Tim -o- ...from me.
don't forget the movie later -oo- ...from tim.
OK -oo- ...from me.
^D ...end of my input.
$ _
```
Although you can have a two-way conversation using `write`, it’s awfully clumsy. A better way is to use the `talk` utility, which is described next.

### 9.5.3 Interactive Conversations: `talk`

The `talk` utility (Figure 9–13) allows you to have a two-way conversation across a network.

---

**Utility:** `talk userId [ tty ]`

The `talk` command allows you to talk to another user on the network via a split-screen interface. If the user is logged onto more than one terminal, you may choose a particular terminal by supplying a specific tty name.

To talk to someone, type the following at your terminal:

```
$ talk theirUserId@theirHost
```

This causes the following message to appear on their screen:

```
Message from TalkDaemon@theirHost...
```

```
talk: connection requested by yourUserId@yourHost
```

```
talk: respond with: talk yourUserId@yourHost
```

If they agree to your invitation, they’ll type the following at their shell prompt:

```
$ talk yourUserId@yourHost
```

At this point, your screen divides into two portions, one containing your keyboard input, and the other containing the other guy’s. Everything that you type is echoed at the other guy’s terminal, and vice versa. To redraw the screen if it ever gets messed up, type `Control-L`. To quit from talk, press `Control-C`.

To prevent other people from talking to you, use the `mesg` utility.

---

Figure 9–13 Description of the `talk` command.

This is a fun utility that is worth exploring with a friend.

### 9.5.4 Messages to Everyone: `wall`

If you ever have something important to say to the world (or at least to everyone on your local host), `wall` is the way to say it. `wall` stands for write-all, and allows you to broadcast a message as described in Figure 9–14.
In the following example, I sent a one-liner to everyone on the local host (including myself):

```
$ wall...
this is a test of the broadcast system
`^D`
...end of input.
```

Broadcast Message from glass@csservr2 (tty2) at 18:04 ...

```
this is a test of the broadcast system
$ _
```

The `wall` command is most often used by system administrators to send users important, timely information (like “System going down in 5 minutes!”).

## 9.6 Distributing Data

A very basic kind of remote operation is the transmission of files, and once again Linux has several utilities for doing this:

- **rcp** (remote copy) and **scp** (secure copy) allow you to copy files between your local Linux host and another remote Linux or UNIX host.
- **ftp** (file transfer protocol or program) and **sftp** (secure ftp) allow you to copy files between your local Linux host and any other host (possibly non-Linux) that supports FTP (the File Transfer Protocol). **ftp** is thus more powerful than **rcp**.
- **uucp** (unix-to-unix copy) is similar to **rcp**, and allows you to copy files between any two Linux or UNIX hosts.

Subsections that follow describe **rcp**, **scp**, **ftp**, and **sftp**.
9.6.1 Copying Files Between Two Linux or UNIX Hosts: rcp and scp

rcp and scp allow you to copy files between Linux or UNIX hosts (Figure 9–15).

### Utility: rcp -p originalFile newFile
- rcp -pr { fileName }+ directory
- scp -p originalFile newFile
- scp -pr { fileName }+ directory

rcp and scp both allow you to copy files between Linux or UNIX hosts. Both your local host and the remote host must be registered as equivalent machines (as described in the section on Security beginning on page 342). To specify a remote file on host, use the syntax:

host:pathName

If pathName is relative, it’s interpreted as being relative to your home directory on host. The -p option tries to preserve the last modification time, last access time, and permission flags during the copy. The -r option causes any file that is a directory to be recursively copied.

When scp is used, the TCP/IP connection to the remote host is encrypted so that a network sniffer cannot observe the data contained in the packets.

Figure 9–15 Description of the rcp and scp commands.

In the following example, I copied the file “original.txt” from the remote “vanguard” host to a file called “new.txt” on my local “csservr2” host. I then copied the file “original2.txt” from my local host to the file “new2.txt” on the remote host.

$ rcp vanguard:original.txt new.txt ...remote to local.
$ rcp original2.txt vanguard:new2.txt ...local to remote.
$ _

9.6.2 Copying Files Between Non-Linux/UNIX Hosts: ftp and sftp

The File Transfer Protocol is a generic protocol for the transmission of files, and is supported by many machines. You can therefore use it to transfer files from your local Linux host to any other kind of remote host as long as you know the Internet address of the remote host’s ftp server. Users of non-Linux or non-UNIX computers often use ftp for transferring files between Linux or UNIX and their own system. Figure 9–16 gives a brief description of ftp and sftp.
Utility: `ftp -n [hostName]`

`sftp [user@]hostName[:file]`

`ftp` and `sftp` allow you to manipulate files and directories on both your local host and a remote host.

If you supply a remote host name, `ftp` searches the "netrc" file to see if the remote host has a passwordless anonymous `ftp` account. If it does, it uses it to log you into the remote host. Otherwise, it assumes that you have an account on the remote host and prompts you for its username and password. If the login is successful, `ftp` enters its command mode and displays the prompt "ftp>". If you don’t supply a remote host name, `ftp` enters its command mode immediately and you must use the open command to connect to a remote host.

The `-n` option prevents `ftp` from attempting the initial automatic login sequence.

`ftp`’s command mode supports many commands for file manipulation. The most common of these commands are described in Figure 9–17. You may abort file transfers without quitting `ftp` by pressing Control-C.

`sftp` opens a secure connection to the remote host and transfers files via encrypted TCP/IP packets. Once a secure connection is established, if `file` is not specified, `sftp` enters an interactive mode similar to `ftp` where files can be transferred.

Figure 9–16 Description of the `ftp` and `sftp` commands.

Figure 9–17 lists the most useful `ftp` commands that are available from its command mode.

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>;command</code></td>
<td>Executes <code>command</code> on local host.</td>
</tr>
<tr>
<td><code>append</code></td>
<td>Appends the local file <code>localFile</code> to the remote file <code>remoteFile</code>.</td>
</tr>
<tr>
<td><code>ascii</code></td>
<td>Transfers a file as ASCII text (maintains proper text format between machines whose text format may differ). ASCII transfer is the default behavior.</td>
</tr>
<tr>
<td><code>bell</code></td>
<td>Causes a beep to be sounded after every file transfer.</td>
</tr>
<tr>
<td><code>binary</code></td>
<td>Transfers a file exactly as it is with no format changes.</td>
</tr>
<tr>
<td><code>bye</code></td>
<td>Shuts down the current remote host connection and then quits <code>ftp</code>.</td>
</tr>
</tbody>
</table>

Figure 9–17 Commands within the `ftp` program. (Part 1 of 2)
In the following example, I copied “writer.c” from the remote host “vanguard” to my local host, and then copied “who.c” from my local host to the remote host:

```
$ ftp vanguard  ...open ftp connection to "vanguard".
Connected to vanguard.utdallas.edu.
vanguard FTP server (SunOS 5.4) ready.
Name (vanguard:glass): glass  ...login
Password required for glass.
Password:              ...secret!
```
354 Chapter 9 • Networking and the Internet

User glass logged in.
ftp> ls ...obtain directory of remote host.
PORT command successful.
ASCII data connection for /bin/ls (129.110.42.1,4919) (0 bytes).
... ...lots of files were listed here.
uniq
upgrade
who.c
writer.c
ASCII Transfer complete.
1469 bytes received in 0.53 seconds (2.7 Kbytes/s)
ftp> get writer.c ...copy from remote host.
PORT command successful.
ASCII data connection for writer.c (129.110.42.1,4920) (1276 bytes).
ASCII Transfer complete.
local: writer.c remote: writer.c
1300 bytes received in 0.012 seconds (1e+02 Kbytes/s)
ftp> !ls ...obtain directory of local host.
reader.c who.c writer.c
ftp> put who.c ...copy file to remote host.
PORT command successful.
ASCII data connection for who.c (129.110.42.1,4922).
ASCII Transfer complete.
ftp> quit ...disconnect.
Goodbye.
$_

9.7 Distributed Processing

The power of distributed systems becomes clearer when you start moving around the network
and logging into different hosts. Some hosts supply limited passwordless accounts with user-
names like “guest” so those explorers can roam the network without causing any harm, although
this practice is fading away as more people abuse the privilege. These days you almost always
have to have an account on a remote computer in order to login. Three utilities for distributed
access are:

• rlogin and slogin, which allow you to log in to a remote Linux or UNIX host
• rsh and ssh, which allow you to execute a command on a remote Linux or UNIX host
• telnet, which allows you to execute commands on any remote host that has a telnet
  server

Of these, telnet is the most flexible, as other systems in addition to Linux support telnet servers.
These three utilities are described in the following sections.
9.7.1 Remote Logins: rlogin and slogin

To log into a remote host, use \texttt{rlogin} (Figure 9–18).

\begin{center}
\begin{tabular}{|l|}
\hline
Utility: \texttt{rlogin} -ec \{ -l \texttt{userId} \} hostName  \\
\texttt{slogin} -ec \{ -l \texttt{userId} \} hostName \\
\texttt{slogin} userID@hostName \\
\hline
\end{tabular}
\end{center}

\texttt{rlogin} and \texttt{slogin} attempt to log you into the remote host hostName. If you don’t supply a username by using the \texttt{-l} option, your local username is used during the login process.

\texttt{slogin} creates a secure connection to the remote host by encrypting all packets used in the network connection.

If the remote host isn’t set as an equivalent of your local host in your “$HOME/.rhosts” file, you are asked for your password on the remote host. \texttt{slogin} will honor that file as well as information found in $HOME/.shosts on the remote host.

Once connected, your local shell goes to sleep and the remote shell starts to execute. When you’re finished with the remote login shell, terminate it in the normal fashion (usually with a \texttt{Control-D}) and your local shell will then awaken.

There are a few special “escape commands” that you may type that have a special meaning; each is preceded by the escape character, which is a tilde (\texttt{~}) by default. You may change this escape character by following the \texttt{-e} option with the preferred escape character.

Here is a list of the escape commands:

\begin{center}
\begin{tabular}{|l|}
\hline
SEQUENCE & MEANING \\
\hline
\texttt{~} & Disconnect immediately from remote host. \\
\texttt{~susp} & Suspend remote login session. Restart remote login using \texttt{fg}. \\
\texttt{~dsusp} & Suspend input half of remote login session, but still echo output from login session to your local terminal. Restart remote login using \texttt{fg}. \\
\hline
\end{tabular}
\end{center}

\textsc{Figure 9–18}  Description of the \texttt{rlogin} and \texttt{slogin} commands.

In the following example, I logged into the remote host “vanguard” from my local host “csservr2”, executed the \texttt{date} utility, and then disconnected:

\begin{verbatim}
$ rlogin vanguard ...remote login.  
Last login: Tue May 19 17:23:51 from csservr2.utdallas  
vanguard% date ...execute a command on vanguard.  
Thu May 19 18:50:47 CDT 2005
\end{verbatim}
vanguard% ^D ...terminate the remote login shell.
Connection closed.
$ _ ...back home again at csservr2!

9.7.2 Executing Remote Commands: rsh and ssh
If you want to execute just a single command on a remote host, rsh and ssh are much handier
than rlogin and slogin (although they are actually the same program, respectively). Figure 9–19
shows how they work.

Utility: rsh [-l userId] hostName [ command ]
        ssh [-l userId] hostName [ command ]
        ssh userID@hostName [ command ]

rsh and ssh attempt to create a remote shell on the host hostName to execute command. Both
utilities copy standard input to command and copy the standard output and errors from command
to their own standard output and error channels. Interrupt, quit, and terminate signals
are forwarded to command, so you may Control-C a remote command. They terminate imme-
diately after command terminates.

If you do not supply a username by using the -l option, your local username is used dur-
ing the connection. If no command is specified, jsh and ssh start a remote shell.

Quoted metacharacters are processed by the remote host; all others are processed by the
local shell.

A connection created by ssh is encrypted.

Figure 9–19 Description of the rsh and ssh commands.

In the following example, I executed the hostname utility on my local “csservr2” host and
the remote “vanguard” host:

$ hostname ...execute on my local host.
csservr2
$ rsh vanguard hostname ...execute on the remote host.
vanguard
$ _
9.7.3 Remote Connections: telnet

Telnet allows you to communicate with any remote host on the Internet that has a telnet server. Figure 9-20 describes how it works:

**Utility: telnet [ host [ port ] ]**

Telnet establishes a two-way connection with a remote port. If you supply a hostname but not a port specifier, you are automatically connected to a telnet server on the specified host, which typically allows you to login to the remote machine. If you don’t even supply a hostname, telnet goes directly into command mode (in the same fashion as ftp).

What happens after the connection is complete depends on the functionality of the port you’re connected to. For example, port 13 of any Internet machine will send you the time of day and then disconnect, whereas port 7 will echo (“ping”) back to you anything that you enter from the keyboard.

To enter command mode after you’ve established a connection, press the sequence Control-, which is the telnet escape sequence. This causes the command-mode prompt to be displayed, which accepts commands including the following:

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>Close current connection.</td>
</tr>
<tr>
<td>open host [ port ]</td>
<td>Connect to host with optional port specifier.</td>
</tr>
<tr>
<td>quit</td>
<td>Exit telnet.</td>
</tr>
<tr>
<td>z</td>
<td>Suspend telnet.</td>
</tr>
<tr>
<td>?</td>
<td>Print summary of telnet commands.</td>
</tr>
</tbody>
</table>

Therefore, to terminate a telnet connection, press Control-] followed by the command quit.

**Figure 9-20** Description of the telnet command.

In the following example, I used telnet to emulate the rlogin functionality by omitting an explicit port number with the open command:

```
$ telnet
...start telnet.
telnet> ?
...get help.
Commands may be abbreviated. Commands are:
```

close  close current connection
logout  forcibly logout remote user and close the connection
display  display operating parameters
mode  try to enter line-by-line or character-at-a-time mode
open  connect to a site
quit  exit telnet
send  transmit special characters ('send ?' for more)
set  set operating parameters ('set ?' for more)
status  print status information
toggle  toggle operating parameters ('toggle ?' for more)
z  suspend telnet
?  print help information

telnet> open vanguard  ...get a login shell from vanguard.
Trying 129.110.43.128 ... Connected to vanguard.utdallas.edu.
Escape character is '^]'.

SunOS 5.4 (vanguard)

login: glass  ...enter my username.
Password: ...secret!
Last login: Wed May 18 17:22:45 from csservr2.utdalla

Thu May 19 17:23:21 CDT 2005
Erase is Backspace
vanguard% date  ...execute a command.
Thu May 19 17:23:24 CDT 2005
vanguard% ^D  ...disconnect from remote host.
Connection closed by foreign host.
$ _  ...telnet terminates.

You may specify the host name directly on the command line if you like:

$ telnet vanguard  ...specify host name on command line.
Trying 129.110.43.128 ... Connected to vanguard.utdallas.edu.
Escape character is '^]'.

SunOS 5.4 (vanguard)

login: glass  ...enter username, etc...

You may use telnet to try out some of the standard port services that I described earlier in this
chapter. For example, port 13 prints the day and time on the remote host and then immediately disconnects:

$ telnet vanguard 13 ...what's the remote time & day?
Trying 129.110.43.128 ...
Connected to vanguard.utdallas.edu.
Escape character is '^]'.
Thu May 19 17:26:32 2005
Connection closed by foreign host ...telnet terminates.
$

Similarly, port 79 allows you to enter the name of a remote user and obtain finger information:

$ telnet vanguard 79 ...manually perform a remote finger.
Trying 129.110.43.128 ...
Connected to vanguard.utdallas.edu.
Escape character is '^]'.
glass ...enter the username.
Login name: glass Name: Graham Glass
Directory: /home/glass Shell: /bin/csh
No unread mail
No Plan.
Connection closed by foreign host. ...telnet terminates.
$

When system administrators are testing a network, they often use port 7 to check host connections. Port 7 echoes everything that you type back to your terminal, and is sometimes known as a “ping-port”:

$ telnet vanguard 7 ...try a ping.
Trying 129.110.43.128 ...
Connected to vanguard.utdallas.edu.
Escape character is '^]'.
hi ...my line.
hi ...the echo.
there
there
]^] ...escape to command mode.
telnet> quit ...terminate connection.
Connection closed.
$

telnet accepts numeric Internet addresses as well as symbolic names:

$ telnet 129.110.43.128 7 ... vanguard's numeric addr.
Trying 129.110.43.128 ...
Connected to 129.110.43.128.
Escape character is '^]'.

As local networks began to grow larger and to be connected together, an evolution began. First, some companies connected their own LANs together via private connections. Others transferred data across a network implemented on the public telephone network. Ultimately, the network research being funded by the U.S. Government brought it all together.

It may sound hard to believe, but what we know today as “the internet” was almost inevitable. Although it began in a computer lab, and at the time, most thought only high-powered computer scientists would ever use it, the way we stored and used information almost dictated that we find a better way to move information from one place to another.

Now when I watch television and see web page addresses at the end of commercials for mainstream products, I know the Internet has truly reached common usage. Not only do high-tech companies maintain web pages, even cereal companies have web sites. One may argue about the usefulness of some of these sites, but the fact that they exist tells us a great deal about how society has embraced the new technology.

It makes you wonder how we got here and where we might go with it all.

### 9.8.1 In The Beginning—the 1960s

In the 1960s, man was about to reach the moon, society was going through upheavals on several fronts, and technology was changing more rapidly than ever before. The Advanced Research Projects Agency (ARPA) of the Department of Defense (DoD) was attempting to develop a computer network to connect government computers (and some government contractors’ computers) together. As with so many advances in our society, some of the motivation (and funding) came from a government that hoped to leverage an advance for military and/or defensive capability. High-speed data communication might help win a war at some point. Our Interstate Highway system (network) has its roots in much the same type of motivation.

In the 1960s, mainframe computers still dominated computing, and would for some time. Removable disk packs, small cartridge tapes, and compact disc technology were still in the future. Moving data from one of these mainframe computers to another usually required writing the data on a bulky tape device or some large disk device, physically carrying that medium to the other mainframe computer, and loading the data onto that computer. Although this was done, it was extremely inconvenient.

### 9.8.1.1 A Network Connection

Though computer networking was still in its infancy, local networks did exist, and were the inspiration for what would ultimately become the Internet. During 1968 and 1969, ARPA experimented with connections between a few government computers. The basic architecture
was a 50-Kbps dedicated telephone circuit connected to a machine at each site called an Interface Message Processor (IMP). Conceptually, this is not unlike your personal Internet connection today, if you consider that your modem does the job of the IMP (of course, the IMP was a much more complex device). At each site, the IMP then connected to the computer or computers that needed to access the network.

9.8.1.2 The ARPANET
The ARPANET was born in September of 1969 when the first four IMPs were installed at the University of Southern California, Stanford Research Institute, the University of California at Santa Barbara, and the University of Utah. All of these sites had significant numbers of ARPA contractors. The success of the initial experiments between these four sites generated a great deal of interest on the part of ARPA as well as the academic community. Computing would never be the same.

9.8.2 Standardizing the Internet—the 1970s
The problem with the first connections to the ARPANET was that each IMP was, to some degree, custom designed for each site, depending on the operating systems and network configurations of their computer. Much time and effort had been expended to get this network up to four sites. Hundreds of sites would require hundreds of times this much custom work if it were done in the same fashion.

It became clear that if all the computers connected to the network in the same way and used the same software protocols, they could all connect to each other more efficiently and with much less effort at each site. But at this time, different computer vendors supplied their own operating systems with their hardware, and there was very little in the way of standards to help them interact or cooperate. What was required was a set of standards that could be implemented in software on different systems to allow sharing data in a form that different computers could understand.

Although the genesis of standard networking protocols began in the 1970s, it would be 1983 before all members of the ARPANET used them exclusively.

9.8.2.1 The Internet Protocol family
In the early 1970s, researchers began to design the Internet Protocol. The word “internet” was used since it was more generic (at the time) than ARPANET, which referred to a specific network. The word “internet” referred to the generic internetworking of computers to allow them to communicate.

The Internet Protocol is the fundamental software mechanism that moves data from one place to another across a network. Data to be sent is divided into packets, which is the basic data unit used on a digital computer network. IP does not guarantee that any single packet will arrive at the other end or in what order the packets will arrive, but it does guarantee that if the packet arrives, it will arrive unchanged from the original. This may not seem very useful at first, but stay with me for a moment.

9.8.2.2 TCP/IP
Once you can transmit a packet to another computer and know that, if it arrives at all, it will be correct, other protocols can be added “on top of” the basic IP to provide other functionality.
The Transmission Control Protocol (TCP) is the most often used protocol along with IP (used together they are referred to as TCP/IP). As the name might imply, TCP controls the actual transmission of the stream of data packets. TCP adds sequencing and acknowledgement information to each packet, and each end of a TCP “conversation” cooperates to make sure the original data stream is reconstructed in the same order as the original. When a single packet fails to arrive at the other end due to some failure in the network, the receiving TCP software figures this out because the packet’s sequence number is missing. It can contact the sender and have it send the packet again. Alternatively, the sender, having likely not received an acknowledgment for the packet in question, will eventually retransmit the packet on its own, assuming it was not received. If it was received and only the acknowledgement was lost, the receiving TCP software, upon receiving a second copy, will drop it, since it has already received the first one. The receiver will still send the acknowledgement the sending TCP software was waiting for.

TCP is a connection-oriented protocol. An application program opens a TCP connection to another program on the other computer, and they send data back and forth to each other. When they have completed their work, they close down the connection. If one end (or a network break) closes the connection unexpectedly, this is considered an error by the other end.

9.8.2.3 UDP/IP

Another useful protocol that cooperates with IP is the User Datagram Protocol (sometimes semi-affectionately called the Unreliable Datagram Protocol). UDP provides a low-overhead method to deliver short messages over IP, but does not guarantee their arrival. On some occasions, an application needs to send status information to another application (such as a management agent sending status information to a network or systems management application), but the information is not of critical importance. If it does not arrive, it will be sent again later, unless it is unnecessary for each and every instance of the data to be received by the application. Of course, this assumes that any failure would be due to some transient condition and that “next time” the transmission will work. If it fails all the time, it would imply a network problem existed.

In a case like this, the overhead required to open and maintain a TCP connection is more work than is really necessary. You just want to send a short status message. You don’t really care if the other end gets it (since if they don’t, they probably will get the next one) and you certainly don’t want to wait around for it to be acknowledged. So an unreliable protocol fills the bill nicely.

9.8.2.4 Internet Addressing

When an organization is setting up a LAN to be part of the Internet, it requests a unique Internet IP address from the Network Information Center (NIC). The number that is allocated depends on the size of the organization:

- A huge organization, such as a country or very large corporation, is allocated a Class A address—a number that is the first 8 bits of a 32-bit IP address. The organization is then free to use the remaining 24 bits for labeling its local hosts. The NIC rarely allocates these Class A addresses, as each one uses up a lot of the total 32-bit number space.
• A medium-sized organization, such as a mid-size corporation, is allocated a Class B address—a number that is the first 16 bits of a 32-bit IP address. The organization can then use the remaining 16 bits to label its local hosts.

• A small organization is allocated a Class C address, which is the first 24 bits of a 32-bit IP address.

For example, the University of Texas at Dallas is classified as a medium-sized organization, and its LAN was allocated the 16-bit number 33134. IP addresses are written as a series of four 8-bit numbers, with the most significant byte (8 bits) written first. All computers on the UT Dallas LAN therefore have an IP address of the form 129.110.XXX.YYY, where XXX and YYY are numbers between 0 and 255.

9.8.2.5 Internet Applications
Once a family of protocols existed that allowed easy transmission of data to a remote network host, the next step was to provide application programs that took advantage of these protocols. The first applications to be used with TCP/IP were two programs that were already in wide use.

The telnet program was (and still is) used to connect to another computer on the network in order to login and use that computer from your local computer or terminal. This is quite useful for access to high-priced computing resources. Your organization might not have its own supercomputer, but you might have access to one at another site. Telnet lets you remotely login without having to travel to the other site.

The ftp program was used to transfer files back and forth. While ftp is still available today, most people use web browsers or network file systems to move data files from one computer to another.

9.8.3 Re-Architecting and Renaming the Internet—the 1980s
As more universities and government agencies began using the ARPANET, word of its usefulness spread. Soon corporations were getting connected. At first, because of the funding involved, a corporation had to have some kind of government contract in order to qualify. Over time, this requirement was enforced less and less.

With this growth came headaches. The smaller a network is, and the fewer the nodes that are connected, the easier it is to administer. As the network grows, the complexity of managing the whole thing grows as well.

It became clear that the growth rate that the ARPANET was experiencing would soon outgrow the Defense Department’s ability to manage the network.

New hosts were being added at a rate that required modifications to the network host table on a daily basis. This required each ARPANET site to download new host tables every day, if they wished to have up-to-date tables.

In addition, the number of available hostnames was dwindling, since each hostname had to be unique across the entire network.

2. $129 \times 256 + 110 = 33134$
9.8.3.1 Domain Name Service

Enter DNS, the Domain Name Service. DNS and BIND, the Berkeley Internet Name Daemon, proposed the hierarchy of domain naming of network hosts and the method for providing address information to anyone on the network as they requested it.

In the new system, top-level domain names were established, under which each network site could establish a subdomain. The DoD would manage the top-level domains and delegate management of each subdomain to the entity or organization that registered the domain. The DNS/BIND software provided the method for any network site to do a lookup of network address information for a particular host.

Let’s look at a real-world example of how a hostname is resolved to an address. One of the most popular top-level domains is com, so we’ll use that in our example. The DoD maintained the server for the com domain. All subdomains registered in the com domain were known to this server. When another network host needed an address for a hostname under the com domain, it queried the com name server.

If you attempted to make a connection to snoopy.hp.com, your machine would not know the IP address, because there was no information in your local host table for snoopy.hp.com. Your machine would contact the domain name server for the com domain to ask for the address. That server knows only the address for the hp.com name server; it does not need to know everything under that domain. But since hp.com is registered with it, the com name server can query the hp.com name server for the address.3 Once a name server that has authority for the hp.com domain is contacted, it returns an address for snoopy.hp.com to the requestor (or a message that the host does not exist).

Up to this point, every host name on the ARPANET was just a name, like utexas for the ARPANET host at the University of Texas. Under the new system, this machine would be renamed to be a member of the utexas.edu domain. However, this change could not be made everywhere overnight. So for a time, a default domain .arpa was established. By default, all hosts began to be known under this domain (i.e., utexas changed its name to utexas.arpa). Once a site had taken that single step it could more easily become a member of its “real” domain later, since the software implemented domain names.

Once the ARPANET community adopted this system, all kinds of problems were solved. Suddenly, a hostname had to be unique only within a subdomain. HP’s having a machine called snoopy didn’t mean someone at the University of Texas couldn’t also use that name, since snoopy.hp.com and snoopy.utexas.edu were different names. Duplication of names had not been such a big problem when only mainframe computers were connected to the network, but we were quickly approaching the explosion of workstations, and it would have been a huge problem.

3. Two options are available in the protocols. The first is that the requesting machine may be redirected to a “more knowledgeable” host and may then make follow-up requests until it obtains the information it needs. The second is that the original machine may make a single request, and each subsequent machine that doesn’t have the address can make the follow-up request of the more knowledgeable host on behalf of the original host. This is a configuration option in the domain resolution software and has no effect on how many requests are made or the efficiency of the requests.
then. The other big advantage was that a single networkwide host table no longer had to be main-
tained and updated on a daily basis. Each site kept its own local host tables up-to-date, but would simply query the name server when an address for a host at another site was needed. By querying other name servers, you were guaranteed to receive the most up-to-date information.

The top-level domains most often encountered are listed in Figure 9–21.

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>biz</td>
<td>business</td>
</tr>
<tr>
<td>com</td>
<td>commercial</td>
</tr>
<tr>
<td>edu</td>
<td>educational</td>
</tr>
<tr>
<td>gov</td>
<td>governmental</td>
</tr>
<tr>
<td>info</td>
<td>unrestricted (i.e., anything)</td>
</tr>
<tr>
<td>mil</td>
<td>military</td>
</tr>
<tr>
<td>net</td>
<td>network service provider</td>
</tr>
<tr>
<td>org</td>
<td>nonprofit organization</td>
</tr>
<tr>
<td>XX</td>
<td>two-letter country code</td>
</tr>
</tbody>
</table>

Figure 9–21  Common top-level domain names.

For example, the University of Texas at Dallas LAN has been allocated the name “utdallas.edu”. Once an organization has obtained its unique IP address and domain name, it may use the rest of the IP number to assign addresses to the other hosts on the LAN.

You can see what addresses your local DNS server returns for specific hostnames with the host command, available on most Linux systems (Figure 9–22).

```
Utility: host [ hostname | IPAddress ]
```

The host command contacts the local Name Service and requests the IP address for a given hostname. It can also do a reverse lookup, where by specifying an IP address you receive the hostname for that address.

Figure 9–22  Description of the host command.

The host command is most useful for obtaining addresses of machines in your own network. Machines at other sites around the Internet are often behind firewalls, so the address you get back may not be usable directly. However, host is good for finding out if domain names or web servers (machines
that would need to be outside the firewall for the public to access) within domains are valid. You might see the following type of output from `host`:

```
$ host www.hp.com
www.hp.com is a nickname for www.hpgtm.speedera.net
www.hpgtm.speedera.net has address 192.151.52.187
www.hpgtm.speedera.net has address 192.6.234.8
$ host www.linux.org
www.linux.org has address 198.182.196.56
$ _
```

`host` displays the current IP address(es) for the hostname we requested. When a hostname doesn’t exist or the DNS server can’t (or won’t) provide the address, we’d see something like this:

```
$ host xyzzy
Host not found.
$ _
```

9.8.3.2 DoD Lets Go

Like a parent whose child has grown up and needs its independence, the Department of Defense reached a point where its child, the ARPANET, needed to move out of the house and be on its own. The DoD originally started the network as a research project, a proof of concept. The network became valuable, so the DoD continued to run it and manage it. But as membership grew, the management of this network took more and more resources and provided the DoD fewer and fewer payoffs as non-DoD-related entities got connected. It was time for the Department of Defense to get out of the network management business.

In the late 1980s, the National Science Foundation (NSF) began to build NSFNET. NSFNET took a unique approach, in that it was constructed as a “backbone” network to which other regional networks would connect. NSFNET was originally intended to link supercomputer centers.

Using the same types of equipment and protocols as those making up the ARPANET, NSFNET provided an alternative medium with much freer and easier access than the government-run ARPANET. To most except the programmers and managers involved, the ARPANET appears to have mutated into the Internet of today. In reality, connections to NSFNET (and their regional networks) were created and ARPANET connections were severed, but because of the sharing of naming conventions and appearances, the change was much less obvious to the casual user.

The end result was a network that worked (from the user’s point of view) the same as the ARPANET had, but that, as it grew, was made up of many more corporations and nongovernment agencies. More importantly, this new network was not funded by government money; it was surviving on private funding from those using it.

9.8.4 The Web—the 1990s

The 1990s saw the Internet come into popular use. Although it had grown consistently since its inception, it still belonged predominately to computer users and programmers. Two things
happened to spring the Internet on an unsuspecting public: the continued growth of personal computers in the home and one amazingly good idea.

9.8.4.1 The “killer app”

Again, timing played a role in the history of the Internet. The network itself was growing and being used by millions of people but was still not considered mainstream. The more sophisticated home users were getting connected to the Internet via a connection to their employer’s network or a subscription with a company that provided access to the Internet. These companies came to be known as ISPs, Internet Service Providers. In the early 1990s, only a handful of these existed, as only a few people recognized there was a business in providing Internet access to anyone who wanted it.

Then came Mosaic. Mosaic was the first “browser” and was conceived by software designers at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. With Mosaic, a computer user could access information from other sites on the Internet without having to use the complicated and nonintuitive tools that were popular at the time (e.g., *telnet*, *ftp*).

Mosaic was (and browsers in general are) an application that displays a page of information both textually and graphically, as described by a page description language called Hyper-Text Markup Language (HTML). The most revolutionary aspect of HTML was that of a *hyperlink*, a way to link information in one place in a document to other information in another part of the document (or more generically, in another document).

By designing a page with HTML, you could display information and include links to other parts of the page or to other pages at other sites that contained related information. This created a document which could be “navigated” to allow users to obtain the specific information in which they were interested rather than having to read or search the document in a sequential fashion, as was typical at the time.

Almost overnight, servers sprang up across the Internet to provide information that could be viewed by Mosaic. Now, rather than maintaining an anonymous FTP site, a person or organization could maintain publicly accessible information in a much more presentable format. In accessing anonymous FTP sites, the users usually had to know what they were trying to find, or at best, get the README file which would help them find what they wanted. With a server that provided HTML, the users could simply point-and-click and be taken to the page containing the information they sought.

Of course, not all this magic happened automatically. Each site that maintained any information for external users had to set up a server and format the information. But this was not significantly more work than providing the information via anonymous FTP. Early on, as people switched from providing information via FTP-based tools to using web-based tools, the two alternatives were comparable in terms of the amount of effort required to make data available. As sites have become more sophisticated, the work required has increased, but the payoff in presentation has also increased.
Some of the people involved in the early releases of Mosaic later formed Netscape Communications, Inc., where they applied the lessons they had learned from early browser development and produced Netscape, the next generation in browsers. Since then, led by Netscape and Microsoft Internet Explorer, browsers have evolved to be very sophisticated applications, introducing significant advances to both browsing and publishing every year.

9.8.4.2 The Web Versus the Internet

The word “web” means many different things to different people in different contexts and causes much confusion. Before Mosaic and other browsers, there was just the Internet. As we have already seen, this is simply a worldwide network of computers. This in itself can be diagrammed as a web of network connections. But this is not what the word “web” means here.

When Mosaic, using HTML, provided the capability to jump around from one place to another on the Internet, yet another conceptual “web” emerged. Not only is my computer connected to several others, forming a web, but now my HTML document is also connected to several others (by hyperlinks), creating a virtual spider web of information. This is the “web” that gave rise to the terms “web pages” and “web browsing,” and is commonly referred to as “The World Wide Web.”

When someone talks about “the web” today, they may mean the Internet itself or they may mean the web of information available on the Internet. Although not originally intended this way, the nomenclature “the web” and “the Internet” are often used interchangeably today. However, in proper usage, “the web” refers to the information that is available from the infrastructure of “the Internet.”

9.8.4.3 Accessibility

A few ISPs had sprung up even as “the web” was coming into existence. Once the concept of “the web” gained visibility, it seemed that suddenly everyone wanted to get on the Internet. While electronic mail was always usable and remains one of the most talked-about services provided by access to the Internet, web browsing had the visibility and the public relations appeal to win over the general public.

All of a sudden, average people saw useful (or at least fun) things they could get from being connected to the Internet. It was no longer the sole domain of computer geeks. For better or worse, the Internet would change rapidly. More people, more information, and more demand caused great growth in usage and availability. Of course, with more people come more inexperienced people and more congestion. Popularity is always a double-edged sword.

Another factor boosting the general public’s access to the Internet has been the geometric increase in modem speeds. While large companies have direct connections to the Internet, most individuals have dial-up connections over home phone lines requiring modems. When the top modem speed was 2400 bps (bytes per second), which wasn’t all that long ago, downloading a web page would have been intolerably slow. As modem speeds have increased and high-speed digital lines have become economical for home use, it has become much more reasonable to have more than just a terminal connection via a dial-up connection.
Most of these private connections can be had for between $10 and $60 per month, depending on speed and usage, which has also played a part in attracting the general public. A bill for Internet service that is comparable with a cable bill or phone bill is tolerable. The general public likely would not accept a bill that was an order of magnitude higher than other utility bills.

9.8.4.4 Changes in the Internet
As the public has played a larger and larger part in the evolution of the Internet, some of the original spirit has changed.

The Internet was originally developed “just to prove it could be done.” The original spirit of the Internet, especially in its ARPANET days, was that information and software should be free to others with similar interests and objectives. Much of the original code that ran the Internet (the TCP/IP protocol suite and tools such as ftp and telnet, etc.) were given away by the original authors and modified by others who contributed their changes back to the original authors for “the greater good.”

This was probably what allowed the Internet to grow and thrive in its youth. Today, however, business is conducted over the Internet, and much of the data is accessible for a fee. This is not to say everybody is out to do nothing but make money or that making money is bad. But it represents a significant change in the culture of the Internet.

The Internet needed its “free spirit” origins, but now that mainstream society is using the Internet, it is only natural that it would become more economically oriented. Advertising on websites is common, and some websites require each user to pay a subscription fee in order to be able to “login” to gain access to information. Commerce over the Internet (such as online ordering of goods and services, including online information) is expected to continue to grow long into the future.

9.8.4.5 Security
Entire books have been written about Internet security (e.g., [Cheswick, 1994]). In the future, as more commercial activity takes place across the Internet, the needs and concerns about the security of operations across the Internet will only increase.

In general, a single transfer of data is responsible for its own security. In other words, if you are making a purchase, the vendor will probably use secure protocols to acquire purchase information from you (like credit card information).

Four major risks confront an Internet web server or surfer: information copying, information modification, impersonation, and denial of service. Encryption services can prevent copying or modifying information. User education can help minimize impersonation.

The most feared (and ironically, the least often occurring) risk is the copying of information that travels across the network. The Internet is a public network, and therefore information that is sent “in the clear” (not encrypted) can, in theory, be copied by someone between the sender and the recipient. In reality, since information is divided into packets that may or may not travel the same route to their destination, it is often impractical to try to eavesdrop in order to obtain useful information.
Modification of information that is in transit poses the same problem as eavesdropping with the additional problem of making the modification. While not impossible, it is a very difficult task and usually not worth the effort.

Impersonation of a user, either through a login interface or an e-mail message, is probably the most common type of security breach. Users often do not safeguard their passwords. Once another person knows their username and password, he or she can login and have all the same rights and privileges as the legitimate user. Unfortunately, it is easy to send an e-mail message with forged headers to make it appear the message came from another user. Close examination can usually authenticate the header, but this can still lead to confusion, especially if an inexperienced user receives the message. One might also impersonate another network host by claiming to use the same network address. This is known as spoofing. Spoofing is not a trivial exercise, but an experienced network programmer or administrator can pull it off.

A denial-of-service attack occurs when an outside attacker sends a huge amount of information to a server to overload its capability to do its job. The server gets so bogged down that it either becomes unusable or it completely crashes.

9.8.4.6 Copyright
One of the biggest challenges in the development of information exchange on the Internet is that of copyright. In traditional print media, time is required to reproduce information, and proof of that reproduction will exist. In other words, if I reprint someone else’s text without their permission, the copy I create will prove the action. On the Internet, information can be reproduced literally at the speed of light. In the amount of time it takes to copy a file, a copyright can be violated, leaving very little evidence of the action.

9.8.4.7 Censorship
In any environment where information can be distributed, there will be those who want to limit who can gain access to what information. If the information is mine and I want to limit your access to it, this is called my right to privacy. If the information is someone else’s and I want to limit your access to it, this is called censorship.

This is not to say that censorship is bad. As with so much in our society, the idea alone is not the problem but rather the interpretation of the idea. Censorship on the Internet is, to put it mildly, a complex issue. Governments and organizations may try to limit certain kinds of access to certain kinds of materials (often with the best of intentions). The problem is that, since the Internet is a worldwide resource, local laws have very little jurisdiction over the whole of the Internet. How can a law in Nashville be applied to a web server in Sydney? Even if they decide the web server is doing something illegal, who will prosecute?

9.8.4.8 Misinformation
As much of a problem as copyrighted or offensive material may be, much more trouble is caused by information that is simply incorrect. Since there is no information authority that approves and validates information put on the net, anyone can publish anything. This is a great thing for free speech. But humans tend to believe information they see in print. I’ve heard innumerable stories about people acting on information they found on the web that turned out to be misleading or
wrong. How much credence would you give to a rumor you were told by someone you didn’t know? That’s how much you should give to information you pick up off the web when you aren’t sure of the source.

9.8.4.9 “Acceptable Use”
Many ISPs have an Acceptable Use policy you must adhere to in order to use their service. Over time, this may well solve many of the problems the Internet has had in its formative years. Most of these policies basically ask users to behave themselves and refrain from doing anything illegal or abusive to other users. This includes sending harassing e-mail, copying files that don’t belong to you, and so on.

There is a perceived anonymity of users of the Internet. If you send me an e-mail message that I disagree with, it may be difficult for me to walk over to you and yell at you personally. I might have to settle for YELLING AT YOU IN E-MAIL. Because of this, people tend to behave in ways they would not in person. As the Internet and its users grow up, this problem should lessen.

9.9 Using Today’s Internet
In the past, using the Internet meant keeping track of a collection of commands and ftp sites. You had to keep track of the resources as well as the method of accessing them.

Today, almost everything you access on the Internet is web-based—that is, accessible via a web browser. A web browser is a program, much like any other window-based program, with menu buttons, a control area, and a display area. You type in or select a web address, the browser sends the request to the specified computer on the network (either the local network or the Internet), and displays in the window the information that is returned. I won’t go into detail about how to use Netscape, or any other browser, since trying it yourself is the best way to learn about web browsing. In general, all browsers have a place to type in a web address, a way to view your browser’s history (web addresses you’ve previously visited), and buttons to help you move backward or forward in this list. Most allow you to save and/or print information and store web addresses in a list of “bookmarks” so you can return to the site in the future without having to remember and retype the address.

9.9.1 URLs
A web page is what is displayed in a browser window when you type in a particular web address. This addresses is called a URL, Uniform Resource Locator. For example, the URL for the Prentice Hall web site is:

http://www.prenhall.com

The components of a URL are the protocol to use to obtain the web page, the Internet
address or hostname of the computer where the web page resides, an optional port number, and an optional filename. In the case of the URL above, the port number and filename were omitted, so the browser assumed port 80 and requested the default HTML file at the root of the web server document tree.

The most common protocol is **http** (HyperText Transfer Protocol), which is the protocol for accessing HTML information. An encrypted channel, Secure HTTP, specified with **https**, is used for pages or transactions involving confidential information (e.g., credit card numbers). Most web browsers also support the **ftp** protocol, which gives you a GUI-based way of accessing anonymous ftp sites through your browser. If no protocol is specified, most browsers will assume “http://” goes on the front of the URL, so you can usually leave that off when manually typing in a URL.

When you load a particular web page into your browser, you are typically presented with a nicely formatted display containing information and other highlighted text or icons (hyperlinks) that you can click on to be taken to other related web pages, possibly part of the current web site, or possibly managed by a completely different organization. The hyperlink is the fundamental concept at the heart of the World Wide Web. It results in a web of information, each page containing a link to many other pages. Since many web pages containing related information have links to each other, the result is a “web” of links all over the Internet.

### 9.9.2 Web Searches

So now that you have a browser window and can access web sites, how do you find the information you want? I could list thousands of web sites that contain interesting information, but by the time this book is published, many of them might not be available anymore. Rather than just giving you a fish, I’d rather show you how to fish so you can find anything you might need on your own.

There are more than a few web **search engines** on the Internet. These are sites that build and continuously update their database of web pages and keyword indices relating to them. Normally these are free services; the pages that show search results usually have advertising on them that sponsors pay for to support the cost of running the site.

Some common search engines, in alphabetical order, include:

- www.altavista.com
- www.excite.com
- www.google.com
- www.looksmart.com
- www.lycos.com
- www.webcrawler.com
- www.yahoo.com

I have my own preferences and so will you. Your favorite may depend on the speed of the response, the layout of the information, the quality of the findings, the ease with which you can build a query, or some of the other services the site may provide.
Today, there are so many sites on the Internet, the biggest problem with using a search engine is building a specific enough query so that you don’t get thousands of links, most of which aren’t what you really want.

If you are trying to find a “regular company,” you can often get lucky by guessing. A URL like:

http://www.companyname.com

probably works more often than not.

9.9.3 Finding Users and Domains

The NIC provides web page access to the database of registered Internet users and Internet domains. The user database is not every Internet user, it is only users who have registered with the NIC. This usually includes system and network administrators who manage domain information for a site.

The NIC’s web site is:

http://www.internic.net

and their web page can point you to the resources where you can perform all sorts of searches for domain and Internet information.

CHAPTER REVIEW

Checklist

In this chapter, I described:

• the main Linux network concepts and terminology
• utilities for listing users and communicating with them
• utilities for manipulating remote files
• utilities for obtaining remote login shells and executing remote commands
• the history of the Internet
• protocols used on the Internet
• applications that access the Internet
• the Domain Name Service used on the Internet
• the World Wide Web, web browsing, and web searching

Quiz

1. What’s the difference between a bridge, a router, and a gateway?
2. What’s a good way for a system administrator of a multi-user system to tell people about important events?
3. Why is ftp more powerful than rcp?
4. Why is rcp easier to use than ftp?
5. Describe some uses of common ports.
6. What does machine equivalence mean and how can you make use of it?
7. Why does the NIC allocate very few Class A addresses?
8. What is the difference between the http and https protocols?
9. If you were looking for Sun Microsystems’ web page, what address would you try first?
10. What are the two most significant differences between the TCP and UDP protocols?

Exercises

1. Try out rcp and rsh as follows:
   • copy a single file from your local host to a remote host by using rcp
   • obtain a shell on the remote host using rsh and edit the file that you just copied
   • exit the remote shell using exit
   • copy the file from the remote host back to the local host using rcp
   [level: easy]
2. Use telnet to obtain the time of day at several remote host sites. Are the times accurate relative to each other? [level: medium]
3. Connect to the www.internic.net web site and explore it to find out what kinds of services the NIC provides. Look up information about your domain name (or your ISP’s domain name). [level: medium]

Projects

1. Write a shell script that operates in the background on two machines and ensures that the contents of a named directory on one machine is always a mirror image of another named directory on the other machine. [level: hard]
2. Pretend you want to buy the latest CD of your favorite group but you don’t know of an Internet site that sells them (there are many). Do a web search with several keywords (like music, CD, purchase, and the name of the group). See if you find a way to buy the CD. Explore some of the unwanted sites that come up to find out why they satisfied your search, so that you know how to make a better search next time. [level: medium]