CHAPTER

# 2

# CONFIGURING BASIC NETWORKING

**After reading this chapter and completing the exercises, you will be able to:**

- Describe how network interfaces are initialized
- Configure network interfaces using scripts and text-mode utilities
- Configure Linux networking using popular graphical utilities
- Effectively use networking utilities to test a network and troubleshoot networking problems
- Understand the IPX and AppleTalk protocols

In this chapter, you will learn how the networking principles in Chapter 1 are implemented in Linux, and how to configure Linux networking. This configuration can be accomplished using simple command-line tools or any of several graphical tools. Modern Linux distributions typically include at least one graphical tool.

In the second part of this chapter, you will learn about the basic networking utilities—most of which are command-line tools—and how to test and troubleshoot a network using these tools. Finally, you will learn a little more about other protocols that are occasionally used in Linux to connect with other types of networks. These protocols include IPX and AppleTalk.

## Initializing Network Interfaces

When Linux boots up, it will probably recognize your network interface hardware, install the appropriate drivers, and configure the interfaces so they're ready to use. When it doesn't happen this way, you'll have to troubleshoot to find out why. Troubleshooting requires that you know how to identify network hardware and load the drivers.

The first step is to see whether Linux recognized the interface hardware when it booted up. If you're running a Linux kernel earlier than 2.6, you have to look in the `/proc/net/dev` file to see the interfaces. The lines in this file are too long to display in the limited space on this page. Look at the file on your system by entering this command:

```
cat /proc/net/dev
```

You'll see the interface names, such as `lo` and `eth0`, at the start of each of the displayed lines. You can ignore the rest of the lines.

If you're running a 2.6 kernel or later, you can more easily see the network interface names listed in the `/sys/class/net` directory. Here's an example of what you'll see if you display the contents of this directory:

```
eth0 eth1 lo
```

This shows that there are two physical Ethernet interfaces (`eth0` and `eth1`) and a local loopback interface (`lo`). If you know that your computer has a network interface but you don't see its name in one of these files, it means the interface driver is not loaded.

### Interface Names

Network interfaces have default names. The first Ethernet interface is called **`eth0`**, the second is called `eth1`, and so on. Token-Ring interfaces are called `tr0`, `tr1`, and so on. Table 2-1 provides a partial list of networking interfaces used by Linux.

**Table 2-1** Examples of Linux interface names

| Interface Name | Description |
|---|---|
| `eth0` | Ethernet interface |
| `tr0` | Token-Ring interface |
| `lo` | Local loopback interface |
| `ppp0` | Point-to-Point Protocol |
| `slip` | Serial Line Internet Protocol |
| `plip` | Parallel Line Internet Protocol |
| `arc0` | ARCnet interface |
| `fddi0` | FDDI interface |
| `dlci0` | Frame relay interface |
| `ippp0` | Integrated Services Digital Network (ISDN) modem |

You can change the interface names to suit your taste by using the **ip** program. You'll have to be logged on as root to make changes. If you want to rename the `eth0` interface, you must first take the interface down:

```
ip link set eth0 down
```

change its name:

```
ip link set eth0 name inside
```

then bring the renamed interface back up:

```
ip link set inside up
```

You can display information about the renamed interfaces by using either the `ip` or the `ifconfig` programs. Here's an example of using the `ip` program:

```
ip link show inside
2:  inside: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc
pfifo_fast qlen 1000
       link/ether 00:01:02:ed:b3:bb brd ff:ff:ff:ff:ff:ff
```

Here's an example of using the `ifconfig` program:

```
ifconfig inside
inside  Link encap:Ethernet HWaddr 00:01:02:ED:B3:BB
        inet addr:192.168.1.1 Bcast:192.168.1.255
Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:421 errors:0 dropped:0 overruns:0 frame:0
        TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
        RX bytes:49814 (48.6 Kb) TX bytes:1434 (1.4 Kb)
        Interrupt:11 Base address:0x1400
```

Most Linux users and programmers do not rename their interfaces and many do not know that you can. Consequently, some programs that refer to network interfaces by name only recognize the default names.

## Multiple Interfaces

If your computer has only one Ethernet interface, there's no doubt that it will be called `eth0`. If the computer has two or more interfaces, you face the challenge of determining which of the physical connectors belongs to `eth0`, which belongs to `eth1`, and so on. When Linux boots up, it locates the interfaces and loads drivers for them. The order in which the drivers are loaded determines the interface naming.

Normally, you have no choice about which of the interfaces is detected first and, consequently, the names of the physical interfaces. The order in which the interfaces are detected is determined by your computer's Plug and Play logic in the case of PCI interfaces and by hardware settings (base I/O address) in the case of non-Plug and Play interfaces, such as ISA bus cards.

The most reliable way of determining which connectors belong to which interface names is to connect another computer to each interface and see if the computers can ping one another. After you've determined which physical connectors are associated with the interface names and you connect the appropriate cables, they should stay the same when you reboot as long as you do nothing to change the way the interfaces are detected.

Adding a new interface to your computer can cause existing interfaces to be renamed when the system boots up. Whenever you add another interface, you should verify that the connectors are still associated with the same interface names. If you don't, you won't be the first person to spend hours troubleshooting a problem that is solved by moving the cables to the proper connectors.

## Interface Drivers

Network interfaces use device drivers that are implemented as kernel modules. The modules are usually loaded at system start-up but can be loaded or unloaded while Linux is running. You can see which modules are loaded with the **`lsmod`** command. You'll see a list of modules displayed that include those needed to support the interface. Here's an example of a module used to support Intel Gigabit Ethernet interfaces:

```
e1000          76956    2  (autoclean)
```

How do you know that the e1000 module is used by Intel Gigabit Ethernet interfaces? You learn it with experience. Until you have that experience, you should use Internet search engines, such as Google, to find out. Before you use Google, you must know what network interface you have. If you purchased a network interface card and plugged it in to your computer, you probably know the manufacturer and model. If the interface is built in to your computer, such as a notebook computer, you might not know.

You can see what interfaces are inside your computer by running the `lspci` program. Look for one or more lines that contain the words *Ethernet controller* (in the case of Ethernet). The following is an example of one such line:

```
0000:00:04.0 Ethernet controller: Silicon Integrated Systems
[SiS] SiS900 PCI Ethernet
```

You can then use Google to search for *sis900 ethernet linux*. The SiS900 Ethernet page will probably appear at the top of the list. Click the link and the page will tell you about the SiS900 driver. With this information, you can try loading the driver (module) by using the `modprobe` program:

```
modprobe sis900
```

The **`modprobe`** command automatically handles dependencies. It loads any other modules that are required. The module initializes the interface hardware. If everything is successful, the command prompt returns after the `modprobe` command with no additional output. No news is good news.


TIP

The **insmod** command also loads kernel modules but it doesn't handle dependencies. The `modprobe` command is usually the better choice.



Chances are good that the driver/module will load properly and you'll see the interface name appear in the `/sys/class/net` directory or when you run the **ifconfig** or **ip link** commands. One problem you might have is that the module is not present on your system. The obvious solution is to locate a copy of the module by doing another Google search, downloading it, and installing it on your system.

Kernel modules are located in a directory somewhere below the **/lib/modules** directory. The exact location varies according to your Linux distribution and the kernel version you are using. On a Fedora Core system, the interface modules are in the directory:

```
/lib/modules/2.6.10-1.766_FC3/kernel/drivers/net
```

When you load a kernel module, you don't need to specify the path to the module's directory. The `modprobe` command is aware of the location of the modules on your system. You also don't need to include the ".o" or ".ko" file extension that you see in the modules' filenames. All you need is the name of the module, as shown in the following examples.


NOTE

Interface drivers do not need to be implemented as kernel modules. You can recompile the kernel with the drivers compiled into a monolithic kernel. Directions for doing this are provided in Chapter 4 of the *Guide to Linux Installation and Administration* (Course Technology, ISBN 0-619-13095-4) or the kernel-HOWTO document on *www.tldp.org*.

## ISA Bus Interfaces

Network interfaces that are **ISA bus** cards might require that you specify details such as the I/O address, interrupt number, and DMA channel. You can specify these as command-line parameters with the `modprobe` command. For example, to specify interrupt 15 and I/O address 300:

```
modprobe ne2000 irq=15 io=0x300
```

Alternatively, you can specify them in a configuration file that `modprobe` uses. This file may be called `/etc/modprobe.conf` or `/etc/modules.conf`. Older Linux systems may use the `/etc/conf.modules` file. The file might look like this:

```
alias eth0 3c59x
options 3c59x irq=15 io=0x300
```

The statements that you can include in this file, and their syntax, are documented in the man page for the appropriate configuration file. For a system that uses `/etc/modules.conf`, refer to the `modules.conf` man page.

Some modules allow parameters that are unique to the network hardware. To learn about the parameters that may be applicable to the module you need to use, refer to the kernel