

Chapter 1

Getting Started with Linux

THE FOLLOWING COMPTIA OBJECTIVES ARE COVERED IN THIS CHAPTER:

- ✓ 1.1 Compare and contrast installation sources.
- ✓ 1.7 Configure the boot process including the following (GRUB: /boot/grub/grub.conf, /boot/grub/menu.lst, grub-install, grub).
- ✓ 1.10 Troubleshoot boot issues using the following tools (kernel options, single user mode [including recovering the root user], Rescue—live CDs, DVDs, and USB keys, dmesg).
- ✓ 1.11 Manage devices using the following tools (lsusb, lspci, lsmod, /sys, modprobe, /proc, /etc/modules.conf, /etc/modprobe.conf, Hardware Compatibility List [HCL]).
- ✓ 2.5 Explain the following features and concepts of X11 (Starting and stopping X11, Differences between the X11 client and server, Window managers and display managers (KDM, GDM), Multiple desktops, X11 configuration file (xorg.conf), Terminal emulators (xterm, etc.).



Before you can begin using Linux, you must have a computer that runs the OS. Although computers pre-loaded with Linux exist, many people choose to install the OS themselves. Thus, this book begins with an examination of this topic. You should be aware, however, that Linux isn't a single OS, but rather a family of OSs, so this book doesn't provide a step-by-step description of Linux installation, but only an overview of some of the important concepts and choices available.

This chapter also covers some critical low-level Linux configuration issues. These are the boot process, the graphical user interface (GUI), and Linux hardware management tools. Booting Linux is done with a tool known as a *boot loader*, which can be configured to boot with a variety of options. Linux's GUI is known as the *X Window System* (*X* or *X11* for short). It's different from the GUIs in Microsoft Windows or Mac OS, so it requires special attention. Managing hardware, although in some respects an advanced topic, is very fundamental to the computer's operation, so this chapter concludes with a look at this topic.

Selecting an Installation Method

If you're installing Linux yourself, you have several choices to make even before you sit down at the computer to do the job. The first of these is choosing a *distribution*. This is a collection of all the software that makes up a working computer; several different Linux distributions are available. Once you select a distribution, you may need to choose what installation method to use and how to interact with the installer. Once this is done, you can actually perform the installation.

Choosing a Distribution

Any OS consists of a series of software components that interact with one another to produce a working system. The lowest-level component is known as the *kernel*. The kernel interacts directly with the hardware and manages all the other programs. Technically, the name *Linux* applies only to the Linux kernel; everything else that makes up a working Linux system is outside of the Linux kernel, and most of these tools run on non-Linux systems, such as FreeBSD, Solaris, Mac OS, and even Windows.

The collection of a specific Linux kernel along with specific versions of other tools (servers, shells, e-mail clients, and so on) and configuration files to glue everything

together makes up a distribution. Dozens of Linux distributions exist, but the following handful are the most popular general-purpose distributions:

Debian If you want a distribution that conforms strictly to free software principles, Debian may be what you want. It uses less in the way of flashy GUI front-ends to configuration tools than do many other distributions. See <http://www.debian.org> for more information.

Fedora This distribution is a free variant of Red Hat. It's very popular among hobbyists and small organizations that don't need extensive support. See <http://fedoraproject.org> for more information.

Mandriva Two older distributions, Mandrake and Conectiva, merged to become Mandriva. This distribution, headquartered at <http://www.mandriva.com>, includes variants intended to run from USB flash drives.

Red Hat This distribution is among the oldest of the popular distributions. It is the commercial variant of Fedora, and it's a popular choice among businesses that want official support. See <http://www.redhat.com> for more information.

SUSE This distribution is a general-purpose distribution with both commercial (SUSE; see <http://www.novell.com/linux>) and fully open source (OpenSUSE; see <http://www.opensuse.org/en>) variants.

Ubuntu You can find this distribution at <http://www.ubuntu.com>. It's a variant of Debian that adds more in the way of GUI tools and other features that appeal to less geeky users. It's very popular among hobbyists and home users.

This list is by no means comprehensive; distributions such as Gentoo, Slackware, and others all appeal to certain people. Others are highly specialized—say, for scientific data collection or use on old hardware. If you need to select a Linux distribution, I recommend you peruse the Web sites for the six distributions in the preceding list to decide what to run. Factors you might consider include cost, hardware requirements, paid support, popularity (for ease of finding free support on the Internet), and what your friends and colleagues run.

Most modern distributions, including Fedora, Mandriva, Red Hat, and SUSE, use the RPM Package Manager (RPM; a recursive acronym) for software installation. Debian and Ubuntu use Debian packages for this purpose. This detail is important because it determines what software you'll use for maintaining your software—a topic that's covered in detail in Chapter 7, “Managing Packages and System Backups.” If you don't know what to select, don't worry too much about it, but be sure that any distribution you pick uses one of these two methods. Other methods, such as the Portage system used by Gentoo, may work perfectly well, but they aren't covered by the Linux+ exam. If you can install two distributions, pick one that uses RPMs and another that uses Debian packages, since the Linux+ exam covers both tools.

Selecting an Installation Medium

Linux can be booted and installed from any of several different media—floppy disks, CD-ROMs, network connections, and so on. For both booting and installing files, different media offer different advantages and disadvantages.

The Boot Method

Linux installer programs run within Linux itself. This means that in order to install Linux, you must be able to boot a small Linux system, which is provided by the distribution maintainer. This system is useful only for installing Linux and sometimes for doing emergency maintenance. It typically fits on one or two floppy disks or can boot from a bootable CD-ROM.

Modern BIOSs include options for the selection of a boot medium. Typical choices include the floppy disk, CD-ROM drive, PATA hard disk, SATA hard disk, SCSI hard disk, and USB media. In addition, some network cards include BIOSs that enable a computer to boot from files stored on a server. In theory, any of these media can be used to boot a Linux installer.

Although many boot methods are possible, the most common method by far is to use a bootable CD-ROM or DVD-ROM. If you buy a boxed Linux package, it will come with a bootable disc; or you can download an image from the distribution's Web site, burn it, and boot it.

In the past, floppy disks and booting from DOS or Windows were common methods of launching an installer. These methods have fallen by the wayside, but it's conceivable you'll still run into them, particularly on old, small, or specialized distributions.

Ultimately, the boot method is unimportant, because the same installation programs run no matter what method you choose. Pick the boot method that's most convenient for your hardware and the form of installation medium you've chosen.

Installation Media

The installation medium is the physical form of the source of the Linux files. These are the most common choices:

Optical discs If you buy Linux in a store or from an online retailer, chances are you'll get an optical disc. In fact, most distributions come on multiple CD-ROMs or a single DVD-ROM. Optical disc installations tend to be quick. Most distribution maintainers offer CD-ROM and DVD-ROM image files that you can burn to CD-Rs yourself. To find optical disc image files, check <http://iso.linuxquestions.org>, <ftp://sunsite.unc.edu/pub/linux/distributions> or your chosen distribution's Web or FTP site.

Network If you have a fast network connection, many distributions enable you to install via network connections. You must typically download a small image file for a boot CD-ROM, burn that image to disc, and boot it. This boot disc has only a minimal software set, so when you install from the network, you'll download only what you want to install. The drawback to network installations is that they tend to be slower than installs from CD-ROMs. They require more information from the user, and so they can be more difficult for a new user to get working. They can also fail midway if a network connection goes down or a server stops responding. Network installations may use any of several protocols to transfer files, including FTP, HTTP (Web), SMB (Windows file sharing), and NFS (Unix/Linux file sharing). Precisely which protocols are supported varies from one distribution to another.

Not all distributions support all of these installation options. All mainstream distributions support installation from optical discs, and most support at least one form of network installation. Beyond this, you should check the documentation for the distribution.

As a general rule of thumb, installing from DVD-ROMs makes the most sense on modern systems, since these computers invariably have DVD-ROM drives, and using DVD-ROMs means you won't be asked to swap discs mid-installation, as is likely if you use CD-ROMs. CD-ROMs are useful on older computers, though.

For network installation, FTP and HTTP are common choices for direct installation from remote servers. Both methods work well, but if your network has a firewall that requires use of a proxy server, you may need to enter extra information to have the installer use the proxy server. SMB and NFS are more commonly used on local networks. If you host a distribution's files on a local system, you could use these protocols (or FTP or HTTP) to perform network installations from your local server.

Interacting with the Installer

Most methods of Linux installation require you to make decisions during the process. You may need to tell the system how to partition your hard disk, what your network settings are, and so on. To handle such interactions, distribution maintainers have developed three methods of data entry:

GUI interactions Most Linux distributions employ a GUI installer; the system boots up into a basic GUI display, automatically configures the keyboard and mouse, and then proceeds to offer options. This method of installation is relatively comfortable to most new Linux users; however, the installer may fail to correctly identify the video display and may therefore revert to a text-mode interface.

Text-based interactions Some distributions default to a text-based installer, in which the computer asks questions that require keyboard responses. Typically, you select options from a text-based menu system, so you don't need to know Bash (described in more detail in Chapter 2, "Using Text-Mode Commands") or be otherwise familiar with Linux details. If a GUI installer behaves strangely, or if you prefer a text-based installer, you may be able to enter one even on distributions that use a GUI installer by default. Typically, the boot disc provides a menu early in the boot process with a prompt that explains how to enter the text-mode installer.

Scripted installations If you plan to install Linux many times on identical hardware, a scripted install may be just the ticket. In this type of install, you create a configuration file that describes precisely what you want to do. You then provide this file to the installer, which can do the rest of the job without pestering you with questions. Using a scripted installer requires highly distribution-specific knowledge.

Performing the Installation

Unfortunately, Linux distributions' installers vary substantially in how they work. You should consult your distribution's documentation to learn the details. Generally speaking,

the installer guides you through several steps, each of which sets options for particular OS features:

Language options You'll typically be asked to confirm your language. In fact, you may be asked to do this twice, in order to set the display language and the layout of your keyboard.

Disk partitioning Hard disks are typically split into multiple sections, or *partitions*, which hold data of different types. If you're installing Linux on a blank hard disk, you can probably get by with the default partitioning options. If you need Linux to coexist with another OS or if you have specialized needs, you may want to consult Chapter 6, "Managing Disks," before installing Linux.

Boot options You may be asked to set various boot options, such as enabling the computer to boot another OS in addition to Linux. The upcoming section "Configuring Boot Loaders" describes this topic in more detail.

Network configuration Linux installers typically enable you to set basic network options. Chapter 8, "Configuring Basic Networking," covers this topic in detail. For now, you should know that most networks employ the Dynamic Host Configuration Protocol (DHCP) to set most network options. If your network uses DHCP, setting the DHCP option should get basic network features working. If your network doesn't use DHCP, you'll need to ask your network administrator for advice. If in doubt, you can leave network configuration until later—at least, if you're installing from an optical disc or other local media.

X configuration Modern Linux distributions typically detect your video display hardware and set it up reasonably for use by the X Window System, Linux's GUI environment; however, you may want or need to fine-tune your monitor's resolution or enter other technical data. If this step gives you problems, you can put it off until later. The upcoming section "Configuring X" covers this topic in detail.

Time options You can set the current time and time zone as part of the system installation. One unusual feature of Linux relates to the choice of Universal Coordinated Time (UTC) vs. local time. Linux computes times based on UTC (which is closely related to Greenwich Mean Time, or GMT) and then converts those times to local time based on your time zone. DOS and Windows, however, use local time internally. For this reason, the hardware clock in computers is often set to local time. Linux handles daylight saving time changes more easily if you set your hardware clock to UTC. Thus, you have the option of using either approach. Generally speaking, you should use UTC if Linux is the only OS on the computer or if you only multiboot to other UTC-using OSs, such as FreeBSD or Mac OS. If your system multiboots with Windows, though, you may want to set the hardware clock to local time.

Package selection All distributions install a base set of programs (or *packages*). Some distributions give you options during installation about what additional software to install. This task can sometimes be overwhelming. If you're in doubt, leave the defaults; you can always install software later, as described in Chapter 7.

Account creation You'll usually have to set a password for root, which is the Linux administrative account. Most installers also give you the option of creating one or more user accounts during installation. Account management is covered in more detail in Chapter 5, "Managing Users."

These tasks may be performed in almost any order, and some distributions add more tasks or omit some of them.

Once the basic installation is done, the installer will reboot the computer. With any luck, you'll be greeted by a text-mode or GUI login prompt. You can then enter your username and password to see a working Linux system. Chapter 2 describes the commands used at a text-mode Linux shell. After a GUI login, you'll see a screen from which you can run various programs via a point-and-click interface similar to that in Windows or Mac OS.

The rest of this chapter covers various topics related to booting and hardware configuration. Some of the tasks described in these sections require you to work at a text-mode command shell. You can either log into a text-mode session or locate an option called `xterm`, `Konsole`, `shell`, `terminal`, or something similar from the GUI menu.



If X is running on your computer, you can access a full-screen text-mode session by pressing `Ctrl+Alt+Fn`, where *n* is a number, typically from 1 through 6. To switch back to X, you can press `Alt+F7` (Fedora uses `Alt+F1` for this purpose). These keystrokes switch between *virtual terminals*, which enable you to run multiple text-mode programs, each with its own display. X occupies one virtual terminal of its own.

Configuring Boot Loaders

The Linux kernel is at the heart of a Linux computer; in fact, technically speaking, the kernel *is* Linux—everything else is support programs. Because the kernel must run before Linux is completely booted, the kernel must be loaded into memory in a unique way. A program known as a *boot loader* handles this task. Several boot loaders are available, some of which can boot a Linux kernel directly, and others of which require help to do the job.



This section describes boot loaders for x86 and x86-64 systems using a legacy BIOS. If you're using Linux on another architecture, such as PowerPC or Alpha, the available boot loaders will be different. Consult your distribution's documentation for details.

The Role of the Boot Loader

When it's first powered up, an x86 CPU checks a specific area of memory for code to execute. This code is the BIOS in most systems. You're probably familiar with the BIOS through your computer's BIOS setup screens, which enable you to configure features such as RAM timing and whether or not onboard ports are active. The BIOS also provides code that allows the computer to boot. The BIOS checks the first sector of your hard disk (or of your

floppy disk, CD-ROM, or other disk devices, depending on the BIOS's capabilities and configuration) for a small boot loader program. This program normally resides on the master boot record (MBR) of a hard disk or the boot sector of a floppy disk. The MBR resides on the first sector of a hard disk and controls the boot process. A boot sector is the first sector of a floppy or of a hard disk partition and also controls the boot process. (In the case of a partition's boot sector, it's used after the MBR.)



The BIOS, as it currently exists, is extremely old and limited. A new system, known as the Extensible Firmware Interface (EFI), is poised to replace the BIOS. Intel-based Macintoshes already use EFI, as do a few other systems; however, EFI is not yet common on commodity PCs. If your system uses EFI, its boot process will differ from that described here.

In the case of a PC that runs nothing but Windows, the boot loader in the MBR is hard-coded to check for a secondary boot loader in the active primary partition—that is, a partition that's been marked as holding a bootable OS. This secondary boot loader directly loads the Windows kernel. The approach in Linux is similar, but standard Linux boot loaders are more complex. The Linux Loader (LILO) and the Grand Unified Boot-loader (GRUB) are the most common Linux boot loaders. Both programs enable you to boot the Linux kernel or to redirect the boot process to another OS.

In some cases, a system uses multiple boot loaders. One resides in the MBR, and another resides in the boot sector of an individual disk partition. (OSs on different partitions can each have their own boot sector–based boot loaders.) In this configuration, the MBR-based boot loader is the *primary boot loader*, and the one in a partition's boot sector is a *secondary boot loader*. Some boot loaders work in only one of these positions. It's often possible for a secondary boot loader to redirect the boot process to a different partition, in which case that partition's boot loader becomes the tertiary boot loader, although the configuration is the same as for secondary status.

Available Boot Loaders

Many OSs ship with their own boot loaders, and others are available from third parties. These are some of the most common boot loaders:

LILO This boot loader can directly boot a Linux kernel, and it can function as either a primary or a secondary boot loader. It may also be installed on a floppy disk. LILO can redirect the boot process to non-Linux partitions, and so it can be used to select Linux or Windows in a dual-boot system. Although once very popular, LILO has been largely eclipsed by GRUB as the boot loader of choice for Linux.

GRUB This boot loader is more or less the standard Linux boot loader. GRUB can be installed in the same locations as LILO—a floppy disk, the MBR, or the boot sector of a Linux partition. It can directly load the Linux kernel, as well as some other OS kernels, or it can redirect the boot process to another boot loader to load other OSs, such as Windows.

OS Loader This is one name by which Windows NT/200x/XP/Vista's boot loader goes. Another is NTLDR. This is a secondary boot loader that cannot directly boot Linux, but it can boot a disk file that can contain LILO or GRUB and hence boot Linux indirectly. It's common on some dual-boot installations.

LOADLIN This is an unusual boot loader in that it's neither a primary nor a secondary boot loader. Rather, it's a DOS program that can be used to boot Linux after DOS has already loaded. It's particularly useful for emergency situations because it enables you to boot a Linux kernel using a DOS boot floppy, and you can also use it to pass kernel parameters to influence the booted system's behavior. LOADLIN comes with most Linux distributions, generally in a directory on the main installation CD-ROM.

Many additional third-party boot loaders are available, most of which cannot directly boot a Linux kernel but can boot a partition on which LILO or GRUB is installed. For this reason, this chapter emphasizes configuring GRUB—this boot loader can be used to boot Linux, whether it functions as the primary, secondary, or tertiary boot loader. If you opt to use GRUB as a secondary boot loader, you'll need to consult the documentation for your primary boot loader to learn how to configure it.



On a Linux-only system, there's no need to deal with a third-party boot loader; LILO or GRUB can function as a primary boot loader without trouble on such systems. Third-party boot loaders are most useful when you have two or more OSs installed and particularly when LILO or GRUB has trouble redirecting the boot process to the other OSs, which is rare.

The usual configuration for GRUB is to place it in the MBR. Even in a Linux-only situation, however, it's sometimes desirable to place GRUB in the Linux boot partition. Used in this way, a standard DOS/Windows MBR will boot Linux *if* the Linux boot partition is a primary partition that's marked as active. This configuration can be particularly helpful in Windows/Linux dual-boot configurations because Windows tends to overwrite the MBR at installation. Therefore, putting GRUB in the Linux boot sector keeps it out of harm's way, and you can get GRUB working after installing or reinstalling DOS or Windows by using the DOS or Windows FDISK program and marking the Linux partition as active. If GRUB is on the MBR and is wiped out, you'll need to boot Linux in some other way, such as by using LOADLIN, and then rerun `grub-install` to restore GRUB to the MBR.

Configuring GRUB

GRUB is a collection of several components, including the boot loader code proper, a configuration file, and a set of utilities for installing and manipulating the boot loader code. The boot loader code can read the configuration file, so there's no need to reinstall the boot loader code whenever you change your GRUB configuration. You can even place the configuration file on a non-Linux partition, which can be handy for quickly reconfiguring GRUB from another OS.



GRUB wasn't developed exclusively for Linux. It can be installed from, and used to boot, a wide variety of OSs. Its Web page is <http://www.gnu.org/software/grub>. Most Linux distributions use GRUB Legacy (version 0.97 or earlier). GRUB 2 is currently in development.



Real World Scenario

The 1024-Cylinder Limit

One bane of the PC world that reared its ugly head twice in the 1990s was the so-called *1024-cylinder limit*. This limit is derived from the fact that the x86 BIOS uses a three-number scheme for addressing hard disk sectors. Each sector is identified by a cylinder number, a head number, and a sector number, known collectively as the sector's *CHS address*. The problem is that each of these values is limited in size. The cylinder number, in particular, is allotted only 10 bits and so cannot exceed 2^{10} , or 1,024, values. In conjunction with the limits for sectors and heads, this restricted addressable ATA hard disk size to 504MB in the early 1990s.

When disks larger than 504MB became common, BIOSs were adjusted with *CHS translation* schemes, which allowed them to juggle numbers between cylinders, heads, and sectors. This increased the limit to just under 8GB. A similar scheme abandoned CHS addressing for BIOS-to-disk communications but retained it for BIOS-to-software communications. This was known as *linear block addressing (LBA)* mode.

These limits never affected Linux once it had booted, because Linux could handle more than 10-bit cylinder values, and it could access disks directly using LBA mode. The Linux boot process was limited, however, because LILO (this was pre-GRUB) relied on CHS addressing via the BIOS to boot the kernel. Therefore, the Linux kernel has traditionally had to reside below the 1,024-cylinder mark.

Today, all new BIOSs include support for so-called *extended INT13* calls, which bypass the CHS addressing scheme. These BIOSs support booting an OS from past the 1,024-cylinder mark on a hard disk, but only if the boot loader and OS support this feature. Recent versions of LILO and GRUB support extended INT13 calls, so new Linux distributions can be installed anywhere on a hard disk—if the BIOS supports this feature.

Setting Global GRUB Options

The traditional location for the GRUB configuration file is `/boot/grub/menu.lst`. Fedora, Gentoo, and Red Hat, though, ship with a version of GRUB that uses `/boot/grub/grub.conf` as the configuration file. Whatever the name, the GRUB configuration file has the same basic form, as illustrated in Listing 1.1.

Listing 1.1: Sample menu.lst File

```
default=0
timeout=4
splashimage=(hd0,3)/grub/splash.xpm.gz
title Linux (2.6.29)
    root (hd0,3)
    kernel /bzImage-2.6.29 ro root=/dev/hda9 mem=256M
    boot
title Windows
    rootnoverify (hd0,1)
    chainloader +1
    boot
```



Chapter 3, “Managing Processes and Editing Files,” describes one of many text editors available for Linux.

Because GRUB wasn’t designed exclusively for Linux, it introduces a new way of referring to hard disks and their partitions. Linux refers to hard disks using files in the `/dev` directory, such as `/dev/hda` and `/dev/hda9`. Disks may have names beginning with `hd` (for most PATA disks) or `sd` (for SCSI disks and most SATA disks). Following that string is a letter that refers to the physical disk, so a system could have, for instance, `/dev/sda` and `/dev/sdb`. Partitions on disks are given numbers starting with 1, as in `/dev/sda1` or `/dev/sdb7`.

GRUB uses strings of the form `(hdx,y)` to identify disks, where `x` is a disk number and `y` is a partition number. (The `y` and preceding comma may be omitted to refer to an entire disk or its MBR.) Both the `x` and the `y` are numbered starting from 0, which contrasts with Linux’s numbering partitions starting with 1. Thus, Linux’s `/dev/hda9` is GRUB’s `(hd0,8)`. GRUB doesn’t distinguish between PATA and SCSI disks; `hd0` is the first disk recognized by the BIOS, `hd1` is the second disk, and so on.

The first three lines of Listing 1.1 set global options:

Default OS The `default=0` line tells GRUB to boot the first OS defined in the file by default. If this line read `default=1`, the default would be the second OS, and so on.

Timeout period The `timeout=4` line sets the timeout before booting the default OS to 4 seconds.

Splash image The third line in Listing 1.1 sets a splash image—an image that’s displayed as part of the boot process. Many Linux distributions ship a splash image with their GRUB files to make for a fancier boot loader menu, but you can omit this line if you like. This example uses a GRUB-style hard disk specification to point to the image file. In this case, it’s the `grub/splash.xpm.gz` file on the fourth partition on the first disk (probably `/dev/hda4` or `/dev/sda4`). Depending on where this partition is mounted, that could be `/grub/splash.xpm.gz`, `/boot/grub/splash.xpm.gz`, or some other location.

Setting OS Boot Options

The two OS definitions in Listing 1.1 both begin with the keyword `title`, which provides a label for the OS that's displayed by GRUB when it boots. Subsequent lines may be indented to help distinguish between the OS definitions, but this indentation is optional. Important features of OS definitions include the following:

Root partition The `root` option identifies the GRUB root partition, which is the partition on which the GRUB configuration files reside. If you did *not* set aside a separate partition for `/boot` when you installed Linux, this line will identify the Linux root (`/`) partition, and subsequent file references will be relative to the Linux root partition. If you used a separate `/boot` partition, though, chances are the GRUB root partition will be the Linux `/boot` partition, and GRUB references to files in Linux's `/boot` directory will omit that directory name. Listing 1.1 identifies the GRUB root partition as `(hd0,3)`, which is `/dev/hda4` on a PATA system.



GRUB can read files from several filesystems, including `ext2fs`, `ext3fs`, `ReiserFS`, `FAT`, and `FFS`. You can use any of these filesystems as your GRUB root partition. If you want to use another filesystem, such as `JFS` or `XFS`, as your Linux root partition, you should split off your GRUB root partition from the Linux root partition.

Linux kernel The `kernel` option identifies a Linux kernel or a kernel for certain other Unix-like OSs, such as a GNU Hurd kernel. This reference is relative to the GRUB root partition, as defined by `root`. You can also pass kernel options on this line. Note that the `root` option passed to the Linux kernel identifies the Linux root partition using a Linux device filename, but the `root` option in the GRUB OS definition identifies the GRUB root partition. The two might be the same, but they might not be. In the case of Listing 1.1, they aren't the same—the GRUB root partition is `(hd0,3)`, or `/dev/hda4`, whereas the Linux root partition is `/dev/hda9`. Chances are `/dev/hda4` is the Linux `/boot` partition. The `ro` option passed on the `kernel` line tells the kernel to mount the root partition in read-only mode initially, just as the `read-only` line does in `lilo.conf`.

Root partition without verification The `rootnoverify` option works just like the `root` option, except that it tells GRUB it shouldn't try to access files on the partition in question. It's most often found when booting non-Linux and non-Unix OSs, such as DOS or Windows.

Chain loader The `chainloader +1` line in Listing 1.1 tells the system to load the first sector of the root partition and pass execution to it. This option is common when booting DOS, Windows, or other OSs that place boot loader code in their boot sectors.

Boot directive The `boot` line tells GRUB to actually boot the kernel or boot sector for the OS in this definition. In practice, it can often be omitted.

In order to boot, the GRUB boot loader code must reside in the MBR, the boot partition's boot sector, or a floppy disk. You can do this by using the `grub` utility, which you launch from a text-mode login or `xterm`:

```
# grub
grub> root (hd0,3)
```

```
grub> setup (hd0)
grub> quit
```

These commands set the GRUB root partition (the same as the one defined in your `menu.lst` or `grub.conf` file), install the boot loader code to the MBR of the hard disk (that is, to `hd0`), and exit from the utility. If you want to install the boot loader to a partition, you'd use `setup (hd0,3)` or some other partition identifier rather than using `setup (hd0)`. The `grub-install` program provides a simplified method of performing these steps:

```
# grub-install (hd0)
```

This command installs GRUB to the MBR of the first disk. It should be able to locate the GRUB root partition automatically.

If you installed a distribution that uses GRUB by default, you shouldn't have to perform any of these steps; GRUB should already be installed and working. You might need to reinstall GRUB from an emergency boot system if it becomes corrupted, though, and you might want to replace the installed system if you learn of a serious GRUB bug. If you just want to add a new kernel or OS to your existing GRUB installation, you do *not* need to reinstall the boot loader code; you need only edit the `menu.lst` or `grub.conf` file.

Adding a New Kernel or OS to GRUB

You can add a new kernel or OS to GRUB by copying an existing entry (or using one in Listing 1.1 as a model) and modifying it to suit your needs. When trying a new kernel, don't replace your old kernel; instead, add the new kernel to the `/boot` directory, and add a description of the new kernel to the GRUB configuration file. Remember to change the `title` line so that you can tell your two kernels apart. When you reboot the computer, you should be able to select the new kernel or OS from the list; there's no need to reinstall the GRUB boot loader code using the `grub` or `grub-install` tool.

Naming Kernel Files

A good practice when adding a new kernel is to give it a name that includes its version number or other identifying information. For instance, Listing 1.1's kernel is called `bzImage-2.6.29`, identifying it as a 2.6.29 kernel. If you had such a kernel and wanted to try adding a new feature (say, XFS support), you might call this new kernel `bzImage2.6.29-xfs`. There are no hard-and-fast rules for such naming, so use whatever system you like. As a general rule, though, the base of the name begins with `vmLinux` (for a "raw" kernel file), `mlinuz` (for a kernel compressed with `gzip`), `zImage` (another name for a kernel compressed with `gzip`), or `bzImage` (for a kernel compressed in a way that supports booting larger kernel images). Most distributions use `mlinuz` for their kernels, but locally compiled kernels usually go by the `bzImage` name.

Troubleshooting Boot Problems

Linux normally boots correctly immediately after installation; however, sometimes it doesn't. What's more, boot problems can develop after installation. Knowing how to handle such problems can be a necessary skill. You can interact with GRUB to enter options at boot time and use rescue discs to modify your working system. You can use several different methods to reset the `root` password, should you forget what it is—an extremely important skill, should the need arise! Finally, you can examine boot messages to help identify and debug boot problems.

Setting Kernel Options in GRUB

When GRUB loads, it normally presents a series of kernel and OS options, as defined in the `menu.lst` or `grub.conf` configuration file. You highlight the line with the title of the kernel or OS you want to boot and press the Enter key to boot.

Rather than accept the default options, though, you can edit them: instead of pressing the Enter key, you can type `e` (without pressing Enter) to edit the entry. Once you do this, GRUB presents the options from the configuration file related to the entry you've selected. You can use the keyboard's arrow keys to select a line and then type `e` again to edit that line. Alternatively, you can type `O` or `o` to create a new blank line before or after the selected one. Once you've made any changes you desire, type `b` to boot the entry.



When you edit a GRUB entry during the boot process, the changes you make are temporary. If you want to make permanent changes, you must edit the configuration file as described earlier in “Configuring GRUB.”

Ordinarily, Linux boots into a full multiuser mode in which any authorized user may log in. Sometimes, though, you may want Linux to boot into a simpler mode in which just the root user may access the system. This single-user mode is useful for performing recovery and system maintenance tasks because the processes run by ordinary users, including system software that may run automatically when the system boots in the normal multiuser mode, can interfere with maintenance tasks such as low-level disk checks. To enter single-user mode, follow these steps:

1. Select the Linux entry you want to boot in the GRUB menu.
2. Type `e`, as described earlier, to edit the entry.
3. Select the line that begins with the word `kernel` and type `e` to edit it.
4. Use the arrow keys to move to the end of the line.
5. Press the spacebar, type `single`, and then press the Enter key to accept this change. (Typing `1` will also work.)
6. Type `b` to begin booting the system.

When the computer boots, it will boot directly into a text-mode root Bash prompt. You can use this prompt to change configuration files or otherwise manage the system. When you're done, you can reboot the system or type **telinit 2**, **telinit 3**, or **telinit 5** to start normal system services. (Which number you use depends on your distribution. The `telinit` command is described in more detail in Chapter 4, "Managing System Services.")



If your system boots normally, you may enter single-user mode from a normal boot by typing **telinit 1** at a root Bash prompt. Using GRUB to enter single-user mode is most useful if the system doesn't boot normally.

Using Rescue Discs

If Linux won't boot at all, even into single-user mode, you can use a Linux rescue disc to boot a working Linux installation and use it to recover your normal system. Rescue discs are available on CD-ROMs, DVD-ROMs, USB flash drives, and other removable media. They enable you to boot from the removable medium and make changes to your nonworking installation on your hard disk. Examples of Linux rescue discs include the following:

Your distribution's installation media Most distributions' installation media include some form of rescue mode. Check your documentation, or study the options presented by the boot disc when you boot it. Sometimes distributions provide a separate rescue disc, so you may want to check your distribution's Web site.

Knoppix This package, based at <http://www.knoppix.net>, is a full-fledged Linux installation based on Debian. You can download images that can be burned to CD-R or DVD-R media. Once booted, Knoppix is an unusually complete Linux system, although it's rather sluggish when run from optical media.

SystemRescueCd This package is a dedicated rescue system available as a CD-R image. In many respects it's similar to Knoppix, but it's targeted explicitly as a rescue system, whereas Knoppix serves a broader purpose. SystemRescueCd is based on Gentoo Linux. Check http://www.sysresccd.org/Main_Page for more details.



You can use a rescue disc that's based on a distribution other than the one you use—for instance, Knoppix works fine to recover a Fedora system. Some files' ownership may seem to change when you use a rescue disc, though. Don't try to correct such changes, since they're probably artifacts of different configurations of the rescue disc and your normal installation.

Once your rescue disc boots, you'll probably be presented with a root shell, in either a text-mode session or a GUI login. The rescue disc might or might not detect and mount your normal Linux system, so you may need to perform this task manually. (Chapter 6 describes how to do this.) Once your normal system is mounted, you can edit configuration files to overcome whatever problem is preventing your normal installation from booting.

Resetting the *root* Password

One common problem that may prompt use of single-user mode or a rescue disc is if you've forgotten the root password. If you can boot your system into single-user mode, you can reset the root password using the normal `passwd` utility:

```
# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Chapter 5 describes the `passwd` utility in more detail. For now, know that you can use it to reset the root password; however, you must have root access to do so. Ordinarily, this requires you to have the root password. If you can enter single-user mode, though, you don't need the root password to reset it.

Another approach to resetting the root password is to edit the `/etc/shadow` file, which is described in more detail in Chapter 5. Locate the line that begins with `root` (normally the first line):

```
root:$1$6iwFIKHV$nDk0sd1bW2iGKsaAoxu87t:14329:0:99999:7:::
```

Delete all the text between the first and second colons (`:`) on this line. This string, which looks like gibberish, is the encrypted password. Deleting this string sets a null password so that no password is required to log into the account. Alternatively, you can copy the encrypted password from an account whose password you do remember to the root account's entry.



After setting a null password, particularly on the root account, you should *immediately* log in and set a real password on that account. Leaving a null password in place is a very serious security hole. Resetting the root password by directly editing `/etc/shadow` is handy if you need to boot into a rescue disc for some reason, but it's generally safer to boot into single-user mode and use the `passwd` command.

Examining Boot Messages with *dmesg*

Linux records data on various critical actions as it works. Some of these records end up in log files stored in the `/var/log` directory tree, as described in Chapter 4. One class of data is recorded in a different way, though, and this information is particularly relevant when you want to investigate boot problems. This information is generated by the kernel and is stored in the *kernel ring buffer*. You can examine this buffer with the `dmesg` command.

At a command prompt, type `dmesg` to see the contents of the ring buffer. Ordinarily, the result will be hundreds of lines of text. If you need to study the ring buffer in detail, you may want to employ the `less` pager:

```
$ dmesg | less
```

This command passes the output of `dmesg` through `less`, which is a program that enables you to page forward and backward through a long text file. (Both `less` and the pipe, `|`, are described in more detail in Chapter 2.)

The kernel ring buffer's contents change over time, so if you want to use `dmesg` to debug boot problems, you should use `dmesg` as soon after booting as possible. What you see will most likely seem cryptic at first. I recommend you examine the kernel ring buffer on a working Linux system to get a feel for what it contains. As you learn about Linux, its hardware, and how Linux names its hardware, the `dmesg` output will become less cryptic.

If you know or suspect that a problem is related to a particular hardware device or software configuration, you can search the ring buffer for information on that hardware or software. For instance, if a USB device isn't being detected, you could search for the string `USB` (or `usb`) or search for the name of the device or its driver. You may discover an error message that will point you toward a solution. Even the absence of information may be relevant; for instance, if you find no mention of USB devices, it could mean that the USB drivers aren't being loaded at all.

Configuring X

With Linux installed and booting, you can begin turning your attention to specific configuration tasks. One of the first of these tasks is setting up X. Although X usually works acceptably on a stock installation, sometimes you must change the X software you're using or tweak the X configuration. You should also know how to start and stop X, both temporarily and by setting a default mode.

In addition to the basics of X configuration, you should be familiar with the X environment. Typically, a *desktop environment* runs in X, providing you with an easy way to launch GUI programs, a file manager, and other tools. You should also be familiar with a tool that was mentioned earlier: an `xterm` or similar program that you can use to run text-mode programs in X.

Selecting an X Server

X is a network-enabled GUI system. It consists of an *X server*, which displays information on its local monitor and sends back user input from a keyboard and mouse; and an *X client*, which is a program that relies on the X server for user interaction. Although these two programs frequently run on the same computer, they don't need to do so. Chapter 9, "Configuring Advanced Networking," includes additional information on using X over a network. The

rest of this chapter assumes you'll be running X programs on the same system that runs the X server, but you don't install X differently if you'll be running X programs remotely.

The X server includes the driver for your video card, as well as support for your mouse and keyboard. Therefore, it's important that you know something about your video card when you install and configure your X server.

Determining Your Video Card Chipset

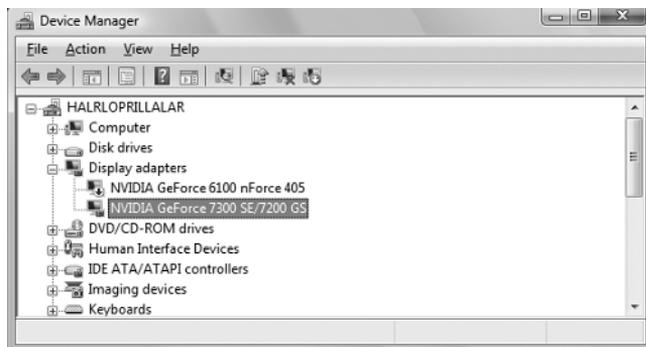
To properly configure X for your system, you must know what video chipset your system uses. Today, three companies—ATI, nVidia, and Intel—dominate the video chipset market, and most video cards and computers are marketed with fairly prominent claims of who made the video chipset. If you're using an oddball or older card, though, you may have trouble finding this information. You have several ways of approaching this problem:

Autodetection Linux can often autodetect the chipset, either during system installation or by running an X configuration tool after installation.

Video card documentation It's worthwhile to check the product's documentation. This documentation might not use the word “chipset,” though; it could use a phrase such as “powered by” or “based on.”

Windows driver report If the computer dual-boots to Windows or if you've just bought a Windows system and intend to convert it to Linux, you can use the Windows Control Panel to find out what video hardware is installed. In Windows Vista, double-click the Device Manager icon in the Control Panel. Click the plus sign next to the Display Adapters item. This will produce a list of the video cards installed in the computer, as shown in Figure 1.1. (Normally, there'll be just one, but Figure 1.1 shows a computer with two video cards: an nVidia GeForce 6100 and an nVidia GeForce 7300.) Double-click the entry for more information; this produces the Properties dialog box for the video card. The driver and manufacturer name may be that of the video card or of the chipset.

FIGURE 1.1 The Windows Device Manager may provide information on the video card hardware.



Linux identification tools The `dmesg` utility, described earlier, may provide clues as to the video card chipset. You can also type `lspci` at a command prompt to obtain identifying information on most of your installed hardware devices, including the video card. Both methods require you to wade through information on nonvideo devices.

One point to keep in mind when identifying the video card chipset is that video cards and video card chipsets are often made by different manufacturers. For instance, nVidia produces chipsets that are used in boards made by ASUS, Biostar, Gigabyte, and others. Linux and X don't care about who made the video card; only the chipset manufacturer is important.



Increasingly, video functions are built into computer motherboards. Nonetheless, it's still common to refer to "video cards" as if they were physically distinct cards, as they once universally were.

Choosing an X Server

All major Linux distributions ship with a free X server. In the past, a server known as XFree86 was common, but most distributions have switched to X.org-X11 instead, because of changes to the XFree86 licensing terms. These two servers are very similar, though; X.org-X11 6.7.0 was based on XFree86 4.3.99. You can learn more about XFree86 at <http://www.xfree86.org>, and X.org-X11 is headquartered at <http://www.x.org>. As I write, the current versions are XFree86 4.8.0 and X.org-X11 7.4.

Linux distributions from 2001 and before used XFree86 3.3.6 or earlier, but more recent distributions use XFree86 4.x or X.org-X11. Some major architectural modifications marked the change to XFree86 4.x, and some configuration files changed with this release. By the time X.org-X11 was forked off the XFree86 project, XFree86 3.3 had become largely obsolete. Thus, I don't cover this old version of XFree86. If you encounter it or must use it because of poor support for an obscure video card in more recent X servers, though, you should be aware that some configuration options changed between XFree86 3.3.6 and 4.0.

Some video card and chipset manufacturers have made XFree86- and X.org-X11-compatible drivers available for their products. Thus, it's worth checking the Web sites maintained by your board and chipset manufacturers to see if drivers are available. This is definitely true if the main XFree86 or X.org-X11 release doesn't include appropriate drivers, and it may be true even if there are drivers—the manufacturers' offerings often offer improved performance, particularly in the realms of 3D and full-motion video acceleration.

XFree86 or X.org-X11 occasionally doesn't support a device at all. You have three choices in this case:

Use the frame buffer device. The Linux kernel has some video drivers of its own. These can be accessed via the *frame buffer* X driver. For this to work, your kernel must include frame buffer support for your video chipset.

Use another X server. It's conceivable that XFree86 will work where X.org-X11 doesn't, or vice versa. In addition, a company called Xi Graphics (<http://www.xig.com>) produces a commercial X server for Linux, known as Accelerated-X. This server occasionally works on hardware that's not supported by XFree86 or X.org-X11, and sometimes it produces better speed.

Replace the hardware. If you have a recalcitrant video card, the final option is to replace it. You may be able to swap with a Windows system that uses a different card, or you may need to buy a new card. Unfortunately, this isn't always an option; you can't replace the video card on a notebook computer, for instance.

Installing an X Server or Driver

Actually installing an X server is usually not very difficult; it's a matter of using your distribution's package management tools to install the software, much as you would any other software (described in Chapter 7). In most cases, this will be done during system installation. You'll have to manually install a server only if you failed to install X during system installation or if you need to install a new server.



X normally comes in several packages. Only one package contains the X server proper; others provide support libraries, fonts, utilities, and so on.

One server package supports all video chipsets. The name of this package varies from one distribution to another, but it's likely to be called XFree86, XFree86-server, xserver-xfree86, or something similar for XFree86; or xorg-x11 or something similar for X.org-X11. Consult Chapter 7 for details of how to locate and install packages.

The main X server program is called X or Xorg, which is usually stored in `/usr/X11R6/bin` or `/usr/bin`. This program is a generic X server. It relies on separate driver modules, which are installed along with the main package in most cases.

If you're using an X driver provided by a video card manufacturer, follow the manufacturer's directions for installing the driver. In most cases you'll be told to run a program that you download from the manufacturer's Web site. Some distributions provide packages with these drivers so you can install them more easily.

Setting Up X

XFree86 is configured through the `XF86Config` file, which is usually located in `/etc` or `/etc/X11`. For XFree86 4.x, this file is sometimes called `XF86Config-4`. X.org-X11 calls its configuration file `xorg.conf`; it's located in the same location and has the same format. (For simplicity, I refer to both files as `xorg.conf` from now on.) Consult your server's documentation if you're using something other than X.org-X11 or XFree86.

Configuring X requires editing the configuration file in any of several ways. You can adjust settings related to input devices (the keyboard and mouse), the video card, and the monitor.

Methods of Configuring X

XFree86 can be configured via either of two methods: by using configuration tools and by configuring manually. Configuration tools prompt you for information or obtain it directly from the hardware and then write the `xorg.conf` file, which is a standard plain-text file like other Linux configuration files. Because this file is relatively complex, it's usually wise to begin with an automatic configuration, even if it's a flawed one. Manual configuration involves opening `xorg.conf` in a text editor and changing its settings using your own know-how. You can use this method to tweak a working configuration for better performance or to correct one that's not working at all. Either way, you may need to configure X, test it, reconfigure X, test it, and so on, for several iterations until you find a configuration that works correctly. (The upcoming section "Starting and Stopping X" describes how to start or restart X to test a new configuration.)

Several utilities can help in X configuration:

The X server The XFree86 or Xorg server itself includes the capacity to query the hardware and produce a configuration file. To do so, type `XFree86 -configure` or `Xorg -configure` when no X server is running. The result should be a file called `/root/XF86Config.new` or `/root/xorg.conf.new`. This file might not produce optimal results, but it is at least a starting point for manual modifications.

Distribution-specific tools Many modern distributions ship with their own custom X configuration tools. These tools frequently resemble the distribution's install-time X configuration tools, which can vary substantially. Increasingly, these tools rely on automatic X detection of hardware and settings, so these tools can be extremely limited; they may give only screen resolution and color depth options, for instance. They can usually be accessed from the desktop's menu system.

In the past, tools such as `xf86config`, `Xconfigurator`, and `XF86Setup` could be used to configure X. These programs are seldom installed on modern systems, though. Therefore, if you need to perform serious changes to your X configuration, chances are you'll need to edit `xorg.conf` in a text editor.

The `xorg.conf` file consists of a number of labeled sections, each of which begins with the keyword `Section`, followed by the section name in quotes, and ends with the keyword `EndSection`. Between these two lines are lines that define features relevant to the configuration of that feature. There may also be comments, which are lines that begin with hash marks (`#`). For instance, here's a section that defines where the computer can find certain critical files:

```
Section "Files"
    RgbPath    "/usr/X11R6/lib/X11/rgb"
    # Multiple FontPath entries are allowed
    FontPath   "/usr/X11R6/lib/X11/fonts/75dpi"
    FontPath   "/usr/X11R6/lib/X11/fonts/Type1"
EndSection
```



If you have a working configuration, be sure to back up `xorg.conf` before modifying it. If you mistakenly delete or modify some critical line, you can easily end up with a system that won't start X, and without a backup, it can be difficult to restore even a partially functioning system.

Many `xorg.conf` sections include `Identifier`, `ModelName`, `VendorName`, or `BoardName` lines. The `Identifier` provides a name for the section that can be used by other sections to refer to the first one. The `ModelName`, `VendorName`, or `BoardName` line, if present, is intended for human consumption, so you can put anything there you like. A `Driver` line, by contrast, points X to a driver for the device. This is extremely important, so you shouldn't change it unless you're positive that the current entry is wrong.

Setting Miscellaneous Options

Some sections of the `xorg.conf` file relate to miscellaneous options or those that require just a handful of lines. Nonetheless, getting these settings right is important to a functioning X system. Specific sections in this category include the following:

Files The `Files` section hosts information on the locations of important files. The entries you're most likely to change relate to the locations of X's fonts. These are handled through the `FontPath` option line. Modern distributions often omit this section and instead rely on default settings. If you need to add fonts to your system, though, you may need to add `FontPath` entries that point to your existing and new font directories.

The keyboard One `InputDevice` section defines the operation of the keyboard in X. This section normally has a `Driver "kbd"` line. In most cases, the default settings (or those set automatically based on your install-time choices) work fine. You may want to adjust the `XkbLayout` or `XkbModel` option to use a different layout or model if these features were set incorrectly. The `AutoRepeat` option sets the delay before keyboard repeat begins and the repeat rate, both in milliseconds (thousandths of a second). This feature is usually overridden in desktop environment configurations, so it may not have any practical effect.

The mouse A second `InputDevice` section, with a `Driver "mouse"` line, defines the mouse. The default settings autodetect the mouse, which works for the vast majority of PS/2 and USB mice. If you use a particularly obscure model, you may need to set `Protocol` and `Device` options, as in `Option "Protocol" "Logitech"` or `Option "Device" "/dev/ttyS1"`.



X programs frequently use the middle button; for instance, text editors use it for pasting text. Therefore, any Linux workstation should be equipped with a genuine three-button mouse rather than a two-button device. Scroll wheels on mice that are so equipped can usually function as a middle button, as well as handling wheel duty. The `Option "Emulate3Buttons" "yes"` option enables you to use a two-button mouse in Linux, but doing so is awkward.

Setting Monitor Options

Some of the trickiest aspects of X configuration relate to the monitor options. You set these in the `Monitor` section, which looks like this:

```
Section "Monitor"
    Identifier "Iiyama"
    ModelName "VisionMaster Pro 450"
    HorizSync 27.0-115.0
    VertRefresh 50.0-160.0
    # My custom 1360x1024 mode
    Modeline "1360x1024" 197.8 \
        1360 1370 1480 1752 \
        1024 1031 1046 1072 -HSync -VSync
EndSection
```

The `HorizSync` and `VertRefresh` lines are extremely critical; they define the range of horizontal and vertical refresh rates that the monitor can accept, in kilohertz (kHz) and hertz (Hz), respectively. Together, these values determine the maximum resolution and refresh rate of the monitor. X won't exceed these limits, since doing so can theoretically damage the monitor. (All monitors made since the mid-1990s have circuitry to protect them from such abuse, so this concern isn't as important as it once was.)

Some X configuration utilities show a list of monitor models or resolution and refresh rate combinations (such as "1024 × 768 at 72 Hz") to obtain this information. This approach is often simpler to handle, but it's less precise than entering the exact horizontal and vertical sync values.

To settle on a resolution, X looks through a series of *mode lines*, which are specified via the `Modeline` option. Computing mode lines is tricky, so I don't recommend you try it unless you're skilled in such matters. The mode lines define combinations of horizontal and vertical timing that can produce a given resolution and refresh rate. For instance, a particular mode line might define a 1024 × 768 display at a 90Hz refresh rate, and another might represent 1024 × 768 at 72Hz.

When asked to produce a given resolution, X searches all the mode lines that accomplish the job, discards those that the monitor can't handle, and uses the remaining mode line that creates the highest refresh rate at that resolution. (If no mode line supports the requested resolution, X drops down to another specified resolution and tries again.)

`Modeline` entries were common in XFree86 3.3.x. Although they're still supported in XFree86 4.x and X.org-X11, these versions of X include standard mode lines that obviate the need for `Modeline` entries in the configuration file unless you want to use an unusual resolution or refresh rate.

Setting Video Card Options

XFree86 4.x and X.org-X11 use driver modules that are stored in separate files from the main X server executable. You must tell the server which driver module to use in

the `xorg.conf` file. In particular, the driver module is set by a line in the Device section, which resembles the following:

```
Section "Device"
    Identifier "On-Board Video"
    VendorName "ATI"
    BoardName "Radeon HD3200"
    Driver "fglrx"
    BusID "PCI:1:5:0"
EndSection
```

The Driver line is the most important one in this section. Driver files reside in an X drivers directory, such as `/usr/X11R6/lib/modules/drivers/` or `/usr/lib/xorg/modules/drivers/`. Most of the drivers' filenames end in `_drv.o`, and if you remove this portion, you're left with the driver name. For instance, `fglrx_drv.o` corresponds to the `fglrx` driver.

The BusID line in this example uniquely identifies the video card by the slot in which it's inserted. (This number is fixed in the case of video hardware built into the motherboard.) Using this line may be necessary if your computer has two video cards, particularly if they're from the same manufacturer.

Many drivers support additional driver-specific options. Consult the `xorg.conf` man page or other driver-specific documentation for details.

Setting Screen Options

The Screen section ties together the other sections. Here's a short example:

```
Section "Screen"
    Identifier "screen1"
    Device "On-Board Video"
    Monitor "Iiyama"
    DefaultDepth 16
    Subsection "Display"
        Depth 8
        Modes "1280x1024" "1024x768" "640x400"
    EndSubsection
    Subsection "Display"
        Depth 16
        Modes "1024x768" "800x600" "640x480"
        Virtual 1280 1024
        ViewPort 0 0
    EndSubsection
EndSection
```

Several key points in this section should be emphasized:

- The `Identifier` specifies an overall configuration. A configuration file can hold multiple `Screen` sections, as described shortly.
- The `Device` and `Monitor` lines point to specific `Device` and `Monitor` sections, respectively.
- The `DefaultDepth` line specifies the number of bits per pixel to be used by default. For instance, the preceding example sets this value to 16, so a 16-bit color depth is used, resulting in 2^{16} , or 65,536, possible colors.
- Each `Subsection` defines a particular display type. They have associated color depths (specified by the `Depth` line) and a series of resolutions (specified by the `Modes` line). The system tries each resolution specified by the `Modes` line in turn, until it finds one that works. There are also various optional parameters, such as `Virtual` (which defines a virtual screen that can be larger than the one that's actually displayed) and `Viewport` (a point within that virtual display at which the initial display is started).

One final section is required: the `ServerLayout` section. This section consists of lines that identify the default `Screen` section and link it to mouse and keyboard definitions. For instance, a typical configuration will include a `ServerLayout` section resembling the following:

```
Section "ServerLayout"
    Identifier   "layout1"
    Screen      "screen1"
    InputDevice "Mouse1"  "CorePointer"
    InputDevice "Keyboard1" "CoreKeyboard"
EndSection
```



Although I describe the `ServerLayout` section last because it ties together all the other sections, it can appear earlier in the file—perhaps even first. The order of sections in the `xorg.conf` file is arbitrary.

Normally, an `xorg.conf` file will have just one `ServerLayout` section, but by passing the `-layout name` parameter to the server program, you can tell the server to use a different `ServerLayout` section, if one is present. You might use this to start X using a different mouse, for instance—say, a USB mouse on a notebook rather than the built-in PS/2 touch pad.

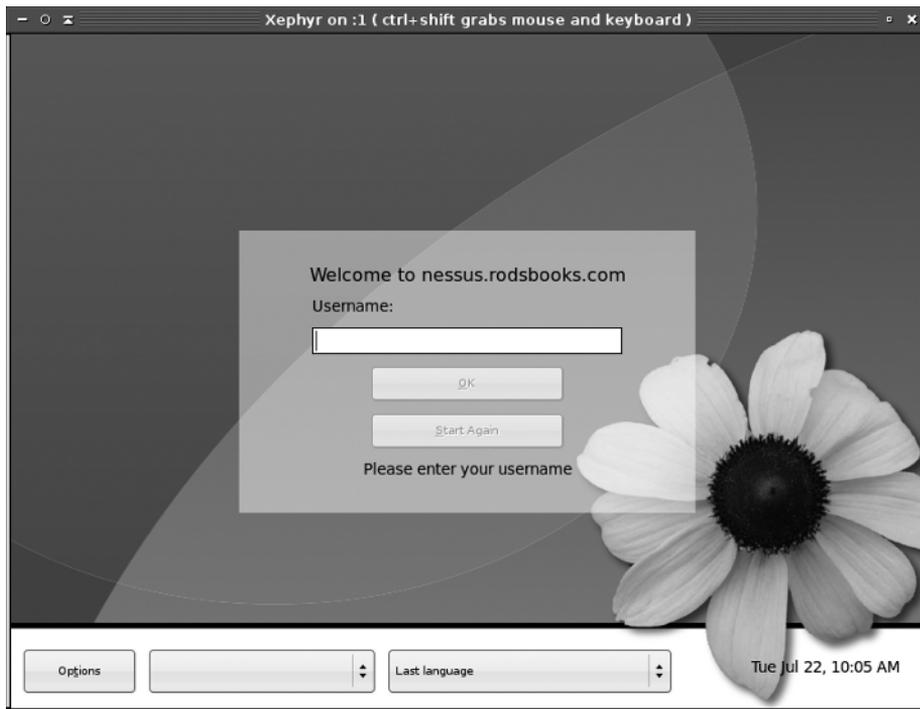
Managing GUI Logins

Linux can boot into a purely text-based mode in which the console supports text-based logins and text-mode commands. This configuration is suitable for a system that runs as a server computer or for a desktop system for a user who dislikes GUIs. Most desktop users, though, expect their computers to boot into a friendly GUI. For such users, Linux supports a login system that starts X automatically and provides a GUI login screen. Configuring and managing this system requires you to understand a bit of how it works, how to run it, and how to change the configuration.

Understanding GUI Logins

X is a network-enabled GUI. This fact has many important consequences, and one of these relates to Linux's GUI login system. This system employs a network login protocol, the *X Display Manager Control Protocol (XDMCP)*. To handle remote logins, an XDMCP server runs on a computer and listens for connections from remote computers' X servers. To handle local logins, an XDMCP server runs on a computer and starts the local computer's X server. The XDMCP server then manages the local X server's display—that is, it puts up a login prompt like that shown in Figure 1.2.

FIGURE 1.2 An XDMCP server manages local GUI logins to a Linux system.



Three XDMCP servers are common on Linux: the X Display Manager (XDM), the KDE Display Manager (KDM), and the GNOME Display Manager (GDM). A few more XDMCP servers are also available, but these three are the most important. As you may guess by their names, KDM and GDM are associated with the KDE and GNOME projects, respectively, but neither limits your choice of desktop environment. Most Linux distributions run either GDM or KDM as the default XDMCP server, but you can change which one your system uses if you don't like the default.

Running an XDMCP Server

Several methods exist to start an XDMCP server. These two are the most common:

- Launching the XDMCP server more or less directly from `init`, via an entry in `/etc/inittab` or its ancillary configuration files. (The `init` program is the first one the kernel launches; it controls the rest of the system startup process through various means.)
- Launching the XDMCP server as part of a runlevel's startup script set, via a SysV startup script. (A runlevel is a set of programs that run concurrently. Several runlevels exist so as to support multiple configurations—such as booting with or without an XDMCP server.)

Chapter 4 describes both `init` and SysV startup scripts in general, so consult it for information about these processes.

Whichever method is used, many distributions configure themselves to run their chosen XDMCP server when they start in runlevel 5 but not when they start in runlevel 3. This is the only difference between these two runlevels in most cases. Thus, changing from runlevel 3 to runlevel 5 starts X and the XDMCP server on many distributions, and switching back to runlevel 3 stops X and the XDMCP server. As described in more detail in Chapter 4, you can change runlevels as root with the `telinit` command:

```
# telinit 5
```

Permanently changing the runlevel requires editing the `/etc/inittab` file and, in particular, its `id` line:

```
id:5:initdefault:
```

Change the number (5 in this case) to the runlevel you want to use as the default.

A few distributions—most notably Gentoo, Debian, and Debian's derivatives (including the popular Ubuntu)—attempt to start an XDMCP server in all runlevels (or don't do so at all). This is done through the use of a SysV startup script called `xdm`, `kdm`, or `gdm`. Thus, you can temporarily start or stop the XDMCP server by running this script and passing it the `start` or `stop` option. To permanently enable or disable the XDMCP server, you should adjust your SysV startup scripts, as described in Chapter 5.

In addition to the question of whether to run an XDMCP server is the question of *which* XDMCP server to run. Most distributions set a default XDMCP server in one way or another. Two common methods exist:

Selection via configuration file Some distributions hide the XDMCP server choice in a configuration file, often in the `/etc/sysconfig` directory. In Fedora, the `/etc/sysconfig/desktop` file, if present, sets the `DISPLAYMANAGER` variable to `XDM`, `KDM`, or `GDM`. In OpenSUSE, `/etc/sysconfig/displaymanager` sets the `DISPLAYMANAGER` variable in a similar way, but using lowercase display manager names.

Selection via SysV script In Debian and derivative distributions, such as Ubuntu, the display manager is set via choice of SysV startup script—use the `gdm` script to use GDM, `kdm`

to use KDM, and so on. By default, only one XDMCP server (and associated SysV startup script) is installed, so if you want to change your XDMCP server, you may need to install your desired server. Chapter 4 describes how to configure specific SysV startup scripts to run automatically.

Unfortunately, distribution maintainers have had a habit of changing the details of how XDMCP servers are launched from time to time, and the settings are often buried in poorly documented configuration files. Thus, you may need to go digging through the files in your `/etc` directory to find the correct setting.

Configuring an XDMCP Server

XDMCP servers, like most programs, can be configured. Unfortunately, this configuration varies from one server to another, although there are some commonalities.

Configuring XDM

XDM is the simplest of the major XDMCP servers. It accepts usernames and passwords but doesn't enable users to perform other actions, such as choose which desktop environment to run. (This must be configured through user login files.)

XDM's main configuration file is `/etc/X11/xdm/xdm-config`. Most distributions ship with a basic `xdm-config` file that should work fine for a local workstation. You can edit this and other XDM configuration files; however, chances are you won't have to do so. One case when you would want to make changes is to enable remote network access to X. This topic is covered in Chapter 10, "Configuring Network Servers."

Configuring KDM

KDM is based partly on XDM and so shares many of its configuration options. Unfortunately, the location of the KDM configuration files is unpredictable; sometimes KDM uses the XDM configuration files, other times they're stored in `/etc/X11/kdm` or `/etc/kde/kdm`, and sometimes they're stored in a truly strange location such as `/usr/share/kde4/config/kdm/kdmrc`.



If you can't find the KDM configuration files, try using your package management tools, described in Chapter 7. Try obtaining lists of files in the `kdebase` package or other likely candidates, and look for the KDM configuration files.

KDM expands on XDM by enabling users to select a session type when they log in, to shut down the computer from the main KDM prompt, and so on. Most of these extra options are set in the `kdmrc` file, which appears in the same directory as the other KDM configuration files. Some of these options override the more common XDM configuration options for the same features.

Configuring GDM

GDM is more of a break from XDM than is KDM. GDM doesn't use the conventional XDM configuration files or similar files. Instead, it uses configuration files that are usually stored in `/etc/X11/gdm` or `/etc/gdm`. The most important of these files is `gdm.conf`, and it has a format

similar to the `kdmrc` file. (Recent versions of GDM use a file called `custom.conf`, which holds only the overrides of default settings stored elsewhere.)

Like KDM, GDM provides extra options over those of XDM. These options include the ability to choose your login environment and shut down the computer. GDM is a bit unusual in that it prompts for the username and only then presents a prompt for the password. (The GDM username prompt was shown in Figure 1.2 earlier in the chapter.) XDM and KDM both present fields for the username and password simultaneously.

Using Window Managers and Desktop Environments

X is a fairly bare-bones environment; it can display windows without borders, it can display text or graphics in those windows, and it can handle some fairly minimal operations beyond that. X does not itself provide menus, buttons, file managers, or other advanced tools. These features are provided by other tools, known as *window managers* and *desktop environments*.

A window manager provides decorative and functional borders around X windows. When you resize or drag a window, it's the window manager that you're using. Most window managers also control the *root window*—that is, the screen as a whole. If you right-click the desktop's background, chances are you'll see a menu pop up; that's a window manager tool. Common window managers include `fvwm`, `twm`, `IceWM`, `Blackbox`, and `Metacity`.

Some minimalist Linux users run a window manager in their X sessions but little else, aside from whatever programs they actively use. Most users, though, run a desktop environment atop the window manager. This is a set of software tools that facilitates launching programs, adjusting user interface settings, and so on. Desktop environments also typically include a file manager, which provides drag-and-drop file manipulation. The most popular Linux desktop environments are the GNU Network Object Model Environment (GNOME), the K Desktop Environment (KDE), and `Xfce`.

If you use KDM or GDM as your XDMCP server, you can select which desktop environment to use when you log in. Look for a menu or option button that enables you to choose the session type. On modern distributions, installing a window manager or desktop environment automatically adjusts the XDM and GDM configurations to present the new environment as a login option.

Once you've logged in, the desktop environment will present menu bars and options that are similar to those of Windows or Mac OS. You'll be able to launch programs by picking them out of menus, you'll be able to access your disks, and so on.

Using Terminal Programs

One of the most important programs you can access from an X session is an `xterm` program or something similar. These programs are sometimes referred to collectively as *terminal programs* or *consoles*. They enable you to run text-mode commands and programs from within a GUI environment. Since so many Linux tools are text-based, knowing how to use a terminal program is a critical skill for any Linux administrator.



The term *terminal program* is often applied to a second type of program. This type of terminal program opens a text-mode data transfer session over a communications device, such as an RS-232 serial port or a modem. In years past, terminal programs of this type were commonly used for communicating with remote computers or for linking two or more nearby computers together. With the advent of modern networks and the Internet, terminal programs of this type have become much less important, so the words *terminal program* now usually refer to *xterm*-type programs.

The main trick in running a terminal program is to find one. Most Linux window managers and desktop environments provide one or more entries in their program lists to launch terminal programs. Check your menus, and particularly any submenus entitled Accessories, System, or System Tools, for entries called Terminal, Konsole, or *xterm*. These are the names of three common terminal programs.

If you can't find an entry for a terminal program, try to find an entry entitled Run, Run Other, or something similar. Such an entry should enable you to run an arbitrary program. Type **terminal**, **konsole**, or **xterm** into its text-entry field to launch a terminal.

Once it's running, a terminal program normally launches a *shell*, which is a Linux tool for entering text-mode commands. (Chapter 2 describes shells in more detail.)

Managing Hardware

Most hardware manufacturers ensure that their products work properly under Windows. Some do the same for Linux, but you're usually on your own when you use Linux. Fortunately, tools and resources exist to help with these tasks. You can check for hardware compatibility before you even buy it, using Internet resources. Once hardware is installed, Linux provides tools to help you identify that hardware and to configure it properly.

Finding Compatible Hardware

When you shop for hardware for use on a Linux computer, you must remember that not all hardware works with Linux. Several resources exist to help in this regard:

The Hardware Compatibility List (HCL) This resource, located at <http://www.linuxquestions.org/hcl>, is the closest thing to a comprehensive source for Linux hardware compatibility. You can use it to check on the compatibility of a specific product or to find products in a category that are known to be compatible.

The OpenPrinting Database This site (http://www.linuxprinting.org/prINTER_list.cgi) maintains a list of printers along with comments on their compatibility with Linux.

The SANE Database The Scanner Access Now Easy (SANE) project maintains a database of scanners, with information on their compatibility with the SANE software, which is the main scanner package for Linux. Check <http://www.sane-project.org> for more details.

The ALSA Project The Advanced Linux Sound Architecture (ALSA) project maintains the mainstream Linux sound drivers. Its Web page, <http://alsa-project.org>, includes notes on compatibility with specific devices.

These databases are maintained by driver developers and end users. As such, they're necessarily incomplete and therefore of limited value. Even limited value is better than no value, though, so you should definitely check these sites before buying hardware or to evaluate the compatibility of hardware you already own.

Another resource you may want to consult is your hardware's manufacturer. Check both the Web site for the company that made the product and, if you know it, the company that made the chipset that drives the product. To Linux, the chipset is what's important, but the hardware's manufacturer may have information or even drivers available for download.

Broadly speaking, products that are most likely to give problems are those that include circuitry that requires specialized drivers—video cards, network cards, scanners, printers, and so on. Some products, by contrast, are extremely standardized and so seldom cause problems. Examples in this category include RAM, hard disks, CD-ROM and DVD-ROM drives, keyboards, mice, and monitors.

Identifying Hardware in Linux

If you've installed Linux on a computer and aren't sure what hardware is available or if you've just installed a device and want to verify that it's accessible, you can use various Linux tools to help identify the hardware. Tools to identify PCI cards and USB devices exist. You can also query the kernel drivers that are loaded and examine a special directory to locate hardware information.

Identifying PCI Devices

The Peripheral Component Interconnect (PCI) standard defines a physical and logical set of parameters that enable plug-in cards to be used in any PCI-supporting computer. PCI devices plug into the computer's motherboard or are built into the motherboard directly. In Linux, the `lspci` command displays information on PCI devices. Table 1.1 summarizes the options to this command.

TABLE 1.1 Options for `lspci`

Option	Effect
-v	Increases verbosity of output. This option may be doubled (-vv) or tripled (-vvv) to produce yet more output.
-n	Displays information in numeric codes rather than translating the codes to manufacturer and device names.

TABLE 1.1 Options for `lspci` (*continued*)

Option	Effect
<code>-nn</code>	Displays both the manufacturer and device names and their associated numeric codes.
<code>-x</code>	Displays the PCI configuration space for each device as a hexadecimal dump. This is an extremely advanced option. Tripling (<code>-xxx</code>) or quadrupling (<code>-xxxx</code>) this option displays information about more devices.
<code>-b</code>	Shows IRQ numbers and other data as seen by devices rather than as seen by the kernel.
<code>-t</code>	Displays a tree view depicting the relationship between devices.
<code>-s [[[[<i>domain</i>]:]<i>bus</i>]:] [<i>slot</i>][.<i>func</i>]</code>	Displays only devices that match the listed specification.
<code>-d [<i>vendor</i>]:[<i>device</i>]</code>	Shows data on the specified device.
<code>-i file</code>	Uses the specified file to map vendor and device IDs to names. (The default is <code>/usr/share/misc/pci.ids</code> .)
<code>-m</code>	Dumps data in a machine-readable form, intended for use by scripts. A single <code>-m</code> uses a backward-compatible format, whereas doubling (<code>-mm</code>) uses a newer format.
<code>-D</code>	Displays PCI domain numbers. These numbers normally aren't displayed.
<code>-M</code>	Performs a scan in bus-mapping mode, which can reveal devices hidden behind a misconfigured PCI bridge. This is an advanced option that can be used only by root.
<code>--version</code>	Displays version information.

Identifying USB Devices

Universal Serial Bus (USB) devices normally attach externally to the computer. You can check to see what USB devices are connected using the `lsusb` command, which is similar in many ways to `lspci`. Table 1.2 summarizes `lsusb` options.

TABLE 1.2 Options for `lsusb`

Option	Effect
<code>-v</code>	Increases verbosity of output
<code>-t</code>	Displays a tree view depicting the relationship between devices
<code>-s [[bus]:][devnum]</code>	Displays only devices that match the listed specification
<code>-d [vendor]:[device]</code>	Shows data on the specified device
<code>-D device</code>	Displays information on the specified <i>device</i> , which is a device file in the <code>/dev</code> directory tree
<code>--version</code> or <code>-V</code>	Displays version information

Note that `lsusb` displays information on both the devices that are attached to your computer and on the USB controller in the computer itself.

Identifying Kernel Drivers

Hardware in Linux is handled by kernel drivers, many of which come in the form of *kernel modules*. These are stand-alone driver files, typically stored in the `/lib/modules` directory tree, that can be loaded and unloaded to provide access to hardware. Typically, Linux loads the modules it needs when it boots, but you may need to load additional modules yourself.

You can learn about the modules that are currently loaded on your system by using `lsmod`, which takes no options and produces output like this:

```
$ lsmod
Module                Size  Used by
isofs                  35820  0
zlib_inflate          21888  1 isofs
floppy                 65200  0
nls_iso8859_1          5568   1
nls_cp437              7296   1
vfat                   15680   1
fat                    49536  1 vfat
sr_mod                19236   0
ide_cd                 42848   0
cdrom                  39080  2 sr_mod,ide_cd
```



The example output for `lsmod` has been edited for brevity. Although outputs this short are possible with certain configurations, they're rare.

The most important column in this output is the first one, labeled `Module`; this column specifies the names of all the modules that are currently loaded. You can learn more about these modules with `modinfo`, as described shortly, but sometimes their purpose is fairly obvious. For instance, the `floppy` module provides access to the floppy disk drive.

The `Used by` column of the `lsmod` output describes what's using the module. All the entries have a number, which indicates the number of other modules or processes that are using the module. For instance, in the preceding example, the `isofs` module (used to access CD-ROM filesystems) isn't currently in use, as revealed by its 0 value; but the `vfat` module (used to read VFAT hard disk partitions and floppies) is being used, as shown by its value of 1. If one of the modules is being used by another module, the using module's name appears in the `Used by` column. For instance, the `isofs` module relies on the `zlib_inflate` module, so the latter module's `Used by` column includes the `isofs` module name. This information can be useful when you're managing modules. For instance, if your system produced the preceding output, you couldn't directly remove the `zlib_inflate` module because it's being used by the `isofs` module; but you could remove the `isofs` module, and after doing so you could remove the `zlib_inflate` module. (Both modules would need to be added back to read most CD-ROMs, though.)



The `lsmod` command displays information only about kernel modules, not about drivers that are compiled directly into the Linux kernel. For this reason, a module may need to be loaded on one system but not on another to use the same hardware because the second system may compile the relevant driver directly into the kernel.

Using the `/proc` Filesystem

Linux uses a special filesystem, `/proc`, to control and provide information about much of the hardware on the computer. Although `lspci`, `lsusb`, `lsmod`, and some other tools provide useful information about specific subsystems, you can use `/proc` to obtain information on still more hardware. For instance, the `/proc/scsi` subdirectory hosts information on SCSI devices (as well as devices that look like SCSI devices, such as most SATA disks), `/proc/cpuinfo` delivers information on your CPU, and `/proc/interrupts` displays the interrupts used by hardware devices.

You may want to peruse your `/proc` filesystem to see what sorts of information it can provide. You can use the `cat` command to display the contents of a file, as in `cat /proc/interrupts`. Be aware that many of the files contained in this directory tree hold extremely technical information that may be mystifying unless you have a deep understanding of the hardware involved.



Don't try to modify the files in `/proc`. Writing to these files can cause your hardware to malfunction. In extreme cases you could wipe out your Linux installation!

Managing Kernel Modules

The `lsmod` command, described earlier, tells you what kernel modules are installed; however, you may need to load kernel modules, remove them, or configure how they operate. To perform these tasks, you must use other tools, such as `insmod`, `modprobe`, and `rmmmod`.

Loading Kernel Modules

Linux enables you to load kernel modules with two programs: `insmod` and `modprobe`. The `insmod` program inserts a single module into the kernel. This process requires you to have already loaded any modules on which the module you're loading relies. The `modprobe` program, by contrast, automatically loads any depended-on modules and so is generally the preferred way to do the job.



In practice, you may not need to use `insmod` or `modprobe` to load modules because Linux can load them automatically. This ability relies on the kernel's module autoloader feature, which must be compiled into the kernel, and on various configuration files, which are also required for `modprobe` and some other tools. Using `insmod` and `modprobe` can be useful for testing new modules or for working around problems with the autoloader, though.

In practice, `insmod` is a fairly straightforward program to use; you type it followed by the module filename:

```
# insmod /lib/modules/2.6.29/kernel/drivers/block/floppy.ko
```

This command loads the `floppy.ko` module, which you must specify by filename. Modules have module names, too, which are usually the same as the filename but without the extension, as in `floppy` for the `floppy.ko` file.

You can pass additional module options to the module by adding them to the command line. Module options are highly module-specific, so you must consult the documentation for the module to learn what to pass. Examples include options to tell an RS-232 serial port driver what interrupt to use to access the hardware or to tell a video card framebuffer driver what screen resolution to use.

Some modules depend on other modules. In these cases, if you attempt to load a module that depends on others and those other modules aren't loaded, `insmod` will fail. When this happens, you must either track down and manually load the depended-on modules or use `modprobe`. In the simplest case, you can use `modprobe` much as you use `insmod`, by passing it a module name:

```
# modprobe floppy
```

As with `insmod`, you can add kernel options to the end of the command line. Unlike `insmod`, you specify a module by its module name rather than its module filename when you use `modprobe`. This helps make `modprobe` easier to use, as does the fact that `modprobe`

automatically loads dependencies. This greater convenience means that `modprobe` relies on configuration files. It also means that you can use options (placed between the command name and the module name) to modify `modprobe`'s behavior, as summarized in Table 1.3.

TABLE 1.3 Options for `modprobe`

Option	Effect
<code>-v</code> or <code>--verbose</code>	This option tells <code>modprobe</code> to display extra information about its operations. Typically, this includes a summary of every <code>insmod</code> operation it performs.
<code>-C filename</code>	The <code>modprobe</code> program uses a configuration file called <code>/etc/modprobe.conf</code> . You can change the file by passing a new file with this option, as in <code>modprobe -C /etc/mymodprobe.conf floppy</code> .
<code>-n</code> or <code>--dry-run</code>	This option causes <code>modprobe</code> to perform checks and all other operations <i>except</i> the actual module insertions. You might use this option in conjunction with <code>-v</code> to see what <code>modprobe</code> would do without loading the module. This may be helpful in debugging problems.
<code>-r</code> or <code>--remove</code>	This option reverses <code>modprobe</code> 's usual effect; it causes the program to remove the specified module and any on which it depends. (Depended-on modules are <i>not</i> removed if they're in use.)
<code>-f</code> or <code>--force</code>	This option tells <code>modprobe</code> to force the module loading even if the kernel version doesn't match what the module expects. This action is potentially dangerous, but it's occasionally required when using third-party binary-only modules.
<code>--show-depends</code>	You can see all the modules on which the specified module depends by using this option. It doesn't install any of the modules; it's purely informative in nature.
<code>-l</code> or <code>--list</code>	This option displays a list of available options whose names match the wildcard you specify. For instance, typing <code>modprobe -l v*</code> displays all modules whose names begin with <code>v</code> . If you provide no wildcard, <code>modprobe</code> displays all available modules. Like <code>--show-depends</code> , this option doesn't cause any modules to be loaded.



Table 1.3 is incomplete. Other `modprobe` options are relatively obscure, so you're not likely to need them often. Consult the `modprobe` man page for more information.

Kernel modules can take options that modify their behavior. These options can be specified in the `/etc/modules.conf` file. (Some distributions use files in `/etc/modules.d` instead.)

Removing Kernel Modules

In most cases, you can leave modules loaded indefinitely; the only harm that a module does when it's loaded but not used is to consume a small amount of memory. (The `lsmod` program shows how much memory each module consumes.) Sometimes, though, you may want to remove a loaded module. Reasons include reclaiming that tiny amount of memory, unloading an old module so you can load an updated replacement module, and removing a module that you suspect is unreliable.

The work of unloading a kernel module is done by the `rmmod` command. This command takes a module name as an option:

```
# rmmod floppy
```

This example command unloads the `floppy` module. You can modify the behavior of `rmmod` in various ways, as summarized by Table 1.4. A few more `rmmod` options exist; consult the `rmmod` man page for details.

TABLE 1.4 Options for `rmmod`

Option	Effect
<code>-v</code> or <code>--verbose</code>	This option tells <code>rmmod</code> to display extra information about its operations.
<code>-f</code> or <code>--force</code>	This option forces module removal even if the module is marked as being in use. Naturally, this is a very dangerous option, but it's sometimes helpful if a module is misbehaving in some way that's even more dangerous. This option has no effect unless the <code>CONFIG_MODULE_FORCE_UNLOAD</code> kernel option is enabled.
<code>-w</code> or <code>--wait</code>	This option causes <code>rmmod</code> to wait for the module to become unused, rather than return an error message, if the module is in use. Once the module is no longer being used (say, after a floppy disk is unmounted if you try to remove the <code>floppy</code> module), <code>rmmod</code> unloads the module and returns. Until then, <code>rmmod</code> doesn't return, making it look like it's not doing anything.

Like `insmod`, `rmmod` operates on a single module. If you try to unload a module that's depended on by other modules or is in use, `rmmod` will return an error message. (The `-w` option modifies this behavior, as described in Table 1.4.) If the module is depended on by other modules, those modules are listed, so you can decide whether to unload them. If you want to unload an entire *module stack*—that is, a module and all those upon which it depends—you can use the `modprobe` command and its `-r` option, as described earlier in “Loading Kernel Modules.”

Summary

Before installing Linux, you should take some time to plan the implementation. This begins with determining which Linux distribution to use, and it continues with planning what installation media you want to use.

After installing Linux, you may need to attend to certain details. One of these is boot loader configuration. Although the installer usually gets this detail correct, particularly for single-OS systems, you may want to tweak the settings or add other OSs to the boot loader. You'll also need to understand this process when you install a new kernel down the road. In case you have problems booting your system, you should know how to troubleshoot boot problems by entering single-user mode, examining `dmesg` output, and using rescue discs.

Another common post-installation configuration detail is getting X working. Again, Linux distributions usually configure X correctly during installation, but you may need to tweak the settings or change them at a later date.

Finally, certain hardware management tools are vital to Linux administrators. You must be able to locate compatible hardware, identify the hardware you have installed, and configure it.

Exam Essentials

Summarize the concept of a Linux distribution. A distribution is a collection of software developed by diverse individuals and groups, bound by an installation routine. Linux distributions can differ in many details, but they all share the same heritage and the ability to run the same programs.

Describe when it's most appropriate to use CD-ROM and network installations. CD-ROM installations are most convenient when installing to systems with poor network connectivity or when you have a CD-ROM and want to install quickly. Network installations are convenient when you are installing several systems simultaneously or when you don't have a Linux CD-ROM or a CD-ROM drive on the target system.

Summarize the x86 boot process. The CPU executes code stored on the BIOS, which redirects the CPU to load and execute a boot loader from the MBR. This boot loader may load the OS kernel or redirect the boot process to another boot loader, which in turn loads the kernel and starts the OS running.

Explain the purpose of single-user mode. In single-user mode, Linux runs without most of the background processes or support for multiuser logins that are hallmarks of a normal Linux boot. Relieving the system of these processes enables you to perform low-level maintenance tasks that might not otherwise be possible.

Describe why you might use an emergency disc. An emergency disc enables you to boot a computer into Linux even when the Linux system installed on the computer is too badly damaged to boot. You can then use the disc-based system to perform data recovery or reconfigure the system on the hard disk.

Determine what video chipset your system uses. Many manufacturers document the video card chipset in their manuals or on the product boxes. You can also check the Microsoft Windows System Control Panel, if the manufacturer did not make the information readily available.

Summarize how X determines the monitor's refresh rate. X uses the monitor's maximum horizontal and vertical refresh rates and a series of fixed mode lines, which define particular timings for various video resolutions. X picks the mode line that produces the highest refresh rate supported by the monitor at the specified resolution.

Explain the purpose of an XDMCP server. XDM, KDM, and GDM are the three main Linux XDMCP servers. Each of them enables users to log into the system with X running, thus bypassing the text-mode login that many users find off-putting.

Summarize some tools for identifying hardware in Linux. The `lspci` and `lsusb` commands display information on PCI and USB hardware, respectively. The `lsmod` command displays the kernel driver modules that are loaded. The `/proc` filesystem is a treasure trove of hardware information, although many of its files contain highly technical information that can be difficult to interpret.

Review Questions

1. In what ways do Linux distributions differ from one another? (Choose all that apply.)
 - A. Package management systems
 - B. Kernel development history
 - C. Installation routines
 - D. The ability to run popular Unix servers
2. Which of the following best describes a typical Linux distribution's method of installation?
 - A. The installation program is a small Linux system that boots from floppy, CD-ROM, or hard disk to install a larger system on the hard disk.
 - B. The installation program is a set of DOS scripts that copies files to the hard disk, followed by a conversion program that turns the target partition into a Linux partition.
 - C. The installation program boots only from a network boot server to enable installation from CD-ROM or network connections.
 - D. The installation program runs under the Minix OS, which is small enough to fit on a floppy disk but can copy data to a Linux partition.
3. Which of the following is an advantage of a GUI installation over a text-based installation?
 - A. GUI installers support more hardware than do their text-based counterparts.
 - B. GUI installers can provide graphical representations of partition sizes, package browsers, and so on.
 - C. GUI installers can work even on video cards that support only VGA graphics.
 - D. GUI installers better test the system's hardware during the installation.
4. What is an advantage of a network installation over a DVD-ROM installation from a downloaded image file?
 - A. A network installation can result in less material downloaded.
 - B. A network installation will proceed more quickly once started.
 - C. A network installation will result in fewer disc swaps.
 - D. A network installation will work even if there's a DHCP error.
5. Where might the BIOS find a boot loader?
 - A. RAM
 - B. /dev/boot
 - C. MBR
 - D. /dev/kmem

6. Which command is used to install GRUB into the MBR of your first ATA hard drive?
 - A. `grub (hd0,1)`
 - B. `grub-install /dev/hda1`
 - C. `lilo /dev/hda`
 - D. `grub-install /dev/hda`
7. Which of the following files might you edit to configure GRUB? (Choose all that apply.)
 - A. `/boot/grub/menu.lst`
 - B. `/etc/grub.conf`
 - C. `/boot/grub/grub.conf`
 - D. `/boot/menu.conf`
8. The string `root (hd1,5)` appears in your `/boot/grub/menu.lst` file. What does this mean?
 - A. GRUB tells the kernel that its root partition is the fifth partition of the first disk.
 - B. GRUB looks for files on the sixth partition of the second disk.
 - C. GRUB looks for files on the fifth partition of the first disk.
 - D. GRUB installs itself in `/dev/hd1,5`.
9. What string would you add to the end of a GRUB kernel line to boot into single-user mode? (Choose all that apply.)
 - A. `1`
 - B. `single`
 - C. `emerg`
 - D. `one`
10. You want to examine the kernel ring buffer to debug a hardware problem. How would you do this?
 - A. Type `ringbuffer` at a command prompt.
 - B. Type `dmesg` at a command prompt.
 - C. Reboot and add the string `ring` to the kernel line in GRUB.
 - D. Install a Token Ring device and examine its output.
11. What is an advantage of using an emergency disc versus using single-user mode?
 - A. An emergency disc may work even if Linux won't boot into single-user mode.
 - B. An emergency disc may work even if you've lost the root password.
 - C. An emergency disc may work even if multiple users are logged in.
 - D. An emergency disc may work even if the CPU is defective.

12. To reset a root password that you've forgotten, you blank the second field in root's entry in `/etc/shadow`. What should you do then?
 - A. Copy `/etc/shadow` so that you don't forget the password again.
 - B. Type **shadow-update** to update the shadow passwords.
 - C. Delete the entry for root in `/etc/passwd`.
 - D. Use `passwd` to set a new password for root.
13. Which of the following is the *most* useful information in locating an X driver for a video card?
 - A. The interrupt used by the video card under Microsoft Windows
 - B. The name of the driver used by the card under Microsoft Windows
 - C. Whether the card uses the ISA, VLB, PCI, or AGP bus
 - D. The name of the video card's manufacturer
14. Which of the following summarizes the organization of the `xorg.conf` file?
 - A. The file contains multiple sections, one for each screen. Each section includes subsections for individual components (keyboard, video card, and so on).
 - B. Configuration options are entered in any order desired. Options relating to specific components (keyboard, video card, and so on) may be interspersed.
 - C. The file begins with a summary of individual screens. Configuration options are preceded by a code word indicating the screen to which they apply.
 - D. The file is broken into sections, one or more for each component (keyboard, video card, and so on). The end of the file has one or more sections that define how to combine the main sections.
15. In what section of `xorg.conf` do you specify the resolution that you want to run?
 - A. In the `Screen` section, subsection `Display`, using the `Modes` option
 - B. In the `Monitor` section, using the `Modeline` option
 - C. In the `Device` section, using the `Modeline` option
 - D. In the `DefaultResolution` section, using the `Define` option
16. Which of the following features do KDM and GDM provide that XDM doesn't?
 - A. An encrypted remote X-based access ability, improving network security
 - B. The ability to accept logins from remote computers, once properly configured
 - C. The ability to select the login environment from a menu on the main login screen
 - D. A login screen that shows the username and password simultaneously rather than sequentially

17. You've installed a new PCI Ethernet card, but it doesn't seem to be working. What can you do to verify that the hardware is visible to Linux?
- A. Type **ping 127.0.0.1** to check connectivity.
 - B. Check that the Ethernet cable is firmly plugged in.
 - C. Type **cat /proc/ethernet** at a command prompt.
 - D. Type **lspci** at a command prompt.
18. An administrator types **lsusb** at a Linux command prompt. What type of information will appear?
- A. Basic identifying information on USB devices, including the USB controllers and all attached devices
 - B. Information on the drivers and detailed capabilities of all USB devices attached to the computer
 - C. A report on the success or failure to load the USB driver stack
 - D. A summary of the amount of data that's been transferred to and from USB devices since the computer was booted
19. Which of the following commands loads a kernel module? (Choose all that apply.)
- A. **rmod**
 - B. **modprobe**
 - C. **lsmod**
 - D. **insmod**
20. You use a USB flash drive and, while the drive is still mounted, type **lsmod**, but you see no entries for kernel modules that you know are required to access a USB flash drive. Why might this be?
- A. The **lsmod** command displays information only on drivers that are in use by the user who typed the command.
 - B. The **lsmod** command displays information only on drivers that are doing work at the moment the command is typed.
 - C. The **lsmod** command displays information only on drivers that are built as modules, not those that are built directly into the kernel.
 - D. The **lsmod** command displays information only on drivers that are used by hardware devices internal to the computer.

Answers to Review Questions

1. A, C. Different Linux distributions use different package management systems and installation routines. Although they may ship with slightly different kernel versions, they use fundamentally the same kernel. Likewise, they may ship with different server collections but can run the same set of servers.
2. A. Most Linux distributions use installation programs written in Linux, not in DOS or Minix. The system usually boots from floppy or CD-ROM, although other boot media (such as hard disk or even network) are possible.
3. B. A bitmapped display, as used by a GUI installer, can be used to show graphical representations of the system's state that can't be done in a text-mode display. Text-based installers actually have an edge in hardware support because they can run on video cards that aren't supported by X.
4. A. When you download a DVD-ROM image file, you're almost certain to download programs you won't install, whereas with a direct network installation, the installer won't bother to download packages it doesn't install. Thus, option A is correct. Although it's conceivable that a network install will be faster than one from a DVD-ROM drive, this is not certain and probably isn't even likely; thus, option B is incorrect. Network installs and installs from a DVD-ROM are both likely to require no disc swaps, so option C is incorrect. A DHCP error refers to an inability to assign a computer an address on the network, so a network install might fail if this occurs, and option D is incorrect.
5. C. The master boot record (MBR) can contain a small boot loader. If more space is required, the boot loader must load a secondary boot loader. Although the boot loader is loaded into RAM, it's not stored there permanently because RAM is volatile storage. Both `/dev/boot` and `/dev/kmem` are references to files on Linux filesystems; they're meaningful only after the BIOS has found a boot loader and run it and lots of other boot processes have occurred.
6. D. You use `grub-install` to install the GRUB boot loader code into an MBR or boot sector. When using `grub-install`, you specify the boot sector on the command line. The MBR is the first sector on a hard drive, so you give it the Linux device identifier for the entire hard disk, `/dev/hda`. Option A specifies using the `grub` utility, which is an interactive tool, and the device identifier shown in option A is a GRUB-style identifier for what would probably be the `/dev/hda3` partition in Linux. Option B is almost correct but installs GRUB to the `/dev/hda1` partition's boot sector rather than the hard disk's MBR. Option C is the command to install LILO to the MBR rather than to install GRUB.
7. A, C. The official GRUB configuration filename is `/boot/grub/menu.lst`; however, some distributions use `/boot/grub/grub.conf` instead. Thus, options A and C are both correct. Options B and D are fictitious.

8. B. The `root` keyword in a GRUB configuration file tells GRUB where to look for files, including its own configuration files, kernel files, and so on. Because GRUB numbers start from 0, `(hd1, 5)` refers to the sixth partition on the second disk. Option A is incorrect because you pass the Linux root partition to the kernel on the kernel line. Options A and C both misinterpret the GRUB numbering scheme. The GRUB installation location is specified on the `grub-install` command line, and `/dev/hd1, 5` isn't a standard Linux device file.
9. A, B. Linux recognizes both `1` and `single` as codes to enter single-user mode in this context. Neither `emerg` nor `one` is a correct answer.
10. B. The `dmesg` command displays the contents of the kernel ring buffer, which holds kernel messages. There is no standard `ringbuffer` command. Adding `ring` to the kernel options when booting will not have the desired effect. Token Ring is a network hardware standard, not a way to examine the kernel ring buffer.
11. A. If Linux is so badly damaged that it won't boot into single-user mode, there's a chance that an emergency disc will give you sufficient access to the computer to fix the problem. Single-user mode bypasses the usual login procedures, so it can work even if you've lost the root password, so option B is incorrect. Both emergency discs and single-user mode require ordinary users to log out, so option C is incorrect. If the CPU is defective, as option D specifies, chances are that neither single-user mode nor an emergency disc will help.
12. D. Blanking the password field as specified in the question results in a null password—no password is required to access the account. This is extremely dangerous, so you should immediately set a new password for the `root` account using `passwd`, as option D specifies. Copying `/etc/shadow` is not required and might pose a security risk, depending on where and how you copy it. There is no standard `shadow-update` utility. Deleting `root` from `/etc/passwd` would likely have disastrous consequences, since this file holds basic account information.
13. B. The driver used under Windows should provide useful information on the video card's chipset, which will help you locate a Linux driver for the card. The video card's manufacturer name might or might not be useful information. If it proves to be useful, you'd also need a model number. The interrupt used by the video card in Windows is irrelevant. The card's bus can narrow the range of possibilities, but it isn't extremely helpful.
14. D. The `xorg.conf` file design enables you to define variants or multiple components and easily combine or recombine them as necessary.
15. A. The `Modeline` option in the Monitor section defines *one* possible resolution, but there are usually several `Modeline` entries defining many resolutions. The `Modeline` option doesn't exist in the `Device` section, however, nor is that section where the resolution is set. There is no `DefaultResolution` section.
16. C. KDM and GDM add many features, one of which is a menu that enables users to select their desktop environment or window manager when they log in rather than specifying it in a configuration file, as option C states. Option A describes one of the advantages of the Secure Shell (SSH) as a remote-access protocol. Option B describes a feature common to all three XDMCP servers. Option D describes the way both KDM and XDM function; GDM is the one that presents username and password fields in series rather than simultaneously.

- 17.** D. The `lspci` command lists all the detected PCI devices, so if the card is installed correctly and working, it should show up in the `lspci` output. Thus, option D is correct. Although `ping` can be a useful network diagnostic tool, option A's use of it doesn't test Ethernet cards, so that option is incorrect. Option B provides sound basic network debugging advice, but it won't help to verify that the card is visible to Linux. There is no `/proc/ethernet` file, so option C won't help.
- 18.** A. The `lsusb` command summarizes the USB devices that are available to the computer, as option A says.
- 19.** B, D. The `modprobe` command loads a module and, if necessary, all those upon which it depends. The `insmod` command loads a module, but only if all its dependencies are met. Thus, options B and D are correct. The `rmmod` command removes a module, and `lsmod` lists the loaded modules, so options A and C are incorrect.
- 20.** C. If a driver is built into the main kernel file, `lsmod` won't display information on it. Thus, if the relevant drivers are built into the kernel, the observed behavior would occur because option C is a correct statement. The `lsmod` command does display information on drivers that are used to service other users' needs, that are loaded but not actively working, and on some types of external hardware devices, contrary to options A, B, and D, respectively.