Understanding Disk Issues

Disks are a critical part of most Linux installations. Therefore, I describe three basic disk issues in this chapter: disk hardware interfaces, disk partitioning, and filesystems. I also describe some of the issues surrounding removable disks, including optical (CD-ROM, DVD-ROM, and Blu-ray) discs.

You can install Linux in a *diskless* configuration, in which a Linux computer boots using files stored on a network server.

Disk Interfaces

Certification Objective

Today, two disk interfaces are common, both of which have already been mentioned:

PATA This interface was very common in the past, but it's fading in popularity. It features wide 40- or 80-pin cables that transfer several bits of data simultaneously—hence the word *parallel* in the name Parallel ATA (PATA). A PATA cable can have up to three connectors—one for the motherboard or disk controller card and two more for up to two hard disks. Alternative names for PATA (or specific variants of it) include Integrated Device Electronics (IDE) or Enhanced IDE (EIDE). The ATA Packet Interface (ATAPI) standard defines a software interface that helps ATA manage devices other than hard disks. Although in some cases the differences between the technologies described by these variant terms are important, today they're often used synonymously.

SATA In 2003, a serial version of the ATA protocol was created, hence Serial ATA (SATA). SATA is more or less software compatible with PATA, but it uses thinner cables that can handle just one hard disk per cable. In 2012, SATA is the dominant disk technology on new computers. An external variant, eSATA, provides high-speed connections to external hard disks.

In addition to these technologies, others exist. The Small Computer System Interface (SCSI) is a parallel interface that was once common on servers and high-end interfaces but is less common today. The Serial Attached SCSI (SAS) is a serial variant that's quite similar to SATA. Both of these technologies are important because ATAPI is modeled after SCSI. The Universal Serial Bus (USB) interface is often used for connecting external disks.

Most modern Linux distributions treat all disks as if they were SCSI disks from a software perspective.

Partitioning a Disk

Certification Objective

You can think of a hard disk as a set of *sectors*, each of which holds a small amount of data—normally 512 bytes, although some disks have larger sectors. The disk hardware itself does little to help organize data on the disk, aside from providing a means to read and write specific sectors. On-disk data management is left up to

the OS. Disk partitions and filesystems are two levels of organization imposed on disks to help manage the data they store.

Partitions are a lot like the drawers in a filing cabinet. Think of a single disk as the main filing cabinet, which is then split up into multiple partitions, much like drawers. This analogy is good as far as it goes, but it has its limits. Unlike filing cabinet drawers, disk partitions can be created in whatever size and quantity are convenient, within the limits of the disk's size. A typical disk has between one and a dozen partitions, although you can create more.

Disk partitions exist to help subdivide the disk into pieces with broadly different purposes, such as partitions for different OSs or for different types of data within an OS. For instance, it's common to create separate partitions for swap space (which is used much like RAM in case you run out of RAM), for user data files, and for the OS itself.

Hard disks and their partitions are frequently represented in diagrams similar to Figure 5.1. This diagram displays partitions as subdivisions of the disk, with partition sizes in the diagram more or less proportional to their true sizes on the disk. Thus, in Figure 5.1 you can see that `/boot` is tiny compared to `/home`. As in the figure, partitions are uninterrupted sections of a disk—that is, `/home`, for instance, is a set of sectors with no other partition carved out of its interior.

Some partitioning tools represent their partitions in a vertical stack rather than a horizontal chain. The principle is the same either way.

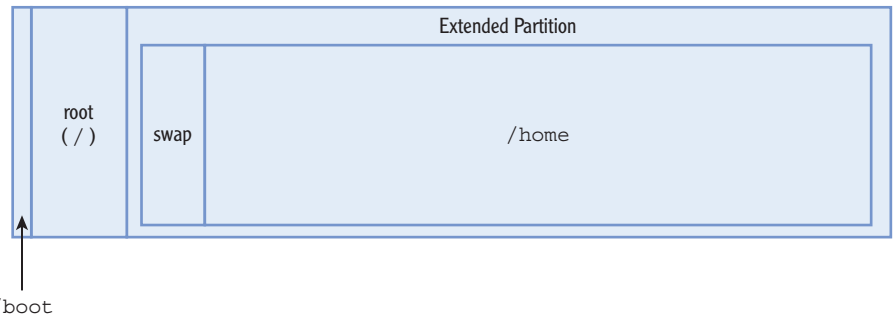

FIGURE 5.1 Disk partitions are often visualized as boxes within a hard disk.

The most common partitioning scheme for *x*86 and *x*86-64 computers has gone by various names over the years, including *master boot record (MBR)*, *MS-DOS*, and *BIOS partitioning*. It supports three types of partitions:

Primary This is the simplest type of partition. A disk can have zero to four primary partitions, one of which may be an extended partition.

Extended This is a special type of primary partition that serves as a placeholder for logical partitions. A disk may have at most one extended partition.

Logical These partitions are contained within an extended partition. In theory, a disk can have billions of logical partitions, thus overcoming the limit of four primary partitions, but in practice you're unlikely to see more than about a dozen of them.

MBR's use of three partition types is awkward and limiting, but inertia has kept it in place for three decades. MBR partitions have a hard limit, though: They can't support disks larger than 2 TiB (tebibytes), assuming 512-byte sectors, which are nearly universal today.

The Globally Unique Identifier (GUID) Partition Table (GPT) is the successor to MBR. GPT supports disks of up to 8 ZiB (zebibytes), which is about 4 billion times as large as MBR's limit. GPT also supports up to 128 partitions by default, with no distinction between primary, extended, and logical partitions. In these respects, GPT is a superior partitioning system to MBR; however, its support varies between OSs. Linux supports both systems quite well. Windows can boot only from MBR when the computer uses the Basic Input/Output System (BIOS), and it can boot only from GPT when the computer is based on the Unified Extensible Firmware Interface (UEFI). Thus, if you dual-boot with Windows, you may need to select your partitioning system with care.

1 TiB is 2^{40} bytes, whereas 1 ZiB is 2^{70} bytes.

Multi-Byte Units

It's common to use prefixes from the International System of Units (SI units)—*kilo* (*k*), *mega* (*M*), *giga* (G), *tera* (T), and so on—in conjunction with *byte* (B) to refer to large quantities of storage space, as in kB, MB, and so on. Technically, these units are defined as base-10 values—*kilo* means 1,000, *mega* means 1,000,000, and so on. In computers, though, base-2 values, such as 2^{10} (1024) and 2^{20} (1,048,576), are often more natural, so the SI units have often (but not always) been used to mean these base-2 values. This practice has led to confusion, since it's not always clear whether base-10 or base-2 units are being used.

To resolve this conflict, the Institute of Electrical and Electronics Engineers (IEEE) defined a new set of prefixes as IEEE-1541. Under this system, new units and prefixes describe base-2 values. The first few of these are:

- A kibibyte (KiB) is 2^{10} (1024) bytes.

(Continues)

MULTI-BYTE UNITS *(Continued)*

- A mebibyte (MiB) is 2^{20} (1,048,576) bytes.
- A gibibyte (GiB) is 2^{30} (1,073,741,824) bytes.
- A tebibyte (TiB) is 2^{40} (1,099,511,627,776) bytes.

In this book, I use IEEE-1541 units when describing features that are best expressed in this system, such as partition table size limits. Most Linux disk utilities use SI and IEEE-1541 units correctly, but which is used depends on the whim of the programs' authors. Be alert to this difference, particularly when dealing with large numbers—note that a tebibyte is almost 10 percent larger than a terabyte!

Several other partitioning systems exist, but you're unlikely to encounter most of them. One possible exception is the Apple Partition Map (APM), which Apple used on its Macintoshes prior to its switch to Intel CPUs.

When it comes to partitioning a disk, Linux supports three families of tools:

`fdisk` family The `fdisk`, `cfdisk`, and `sfdisk` tools are simple text-based partitioning utilities for MBR disks and some more exotic partition table types. These tools work well and provide the means to recover from some disk errors, but their text-based nature can be intimidating to those who are unfamiliar with disk partitioning.

`libparted`-based tools Tools based on the `libparted` library can handle MBR, GPT, and several other partition table types. Some of these tools, such as GNU Parted, are text based, but others, such as GParted, are GUI, and so are likely to be easier for new users to use. Figure 5.2 shows GParted in action. Note how its display mirrors the structure shown in Figure 5.1. Many Linux installers include `libparted`-based partitioning tools that run during system installation.

GPT fdisk family The `gdisk`, `cgdisk`, and `sgdisk` tools are modeled after the `fdisk` family but work with GPT disks. They provide more options for handling GPT than do `libparted`-based tools, but at the cost of friendliness for new users.

I am the author of the GPT fdisk family of partitioning tools.

If you're working with a pre-installed Linux system, you may not need to partition your disk; however, if you ever replace or install a new hard disk, you'll have to partition it before you can use it. You may also need to partition removable disks, although they generally come from the factory pre-partitioned with one big partition.

To partition a disk, you must know the disk's device filename. In Linux, these filenames are normally `/dev/sda`, `/dev/sdb`, and so on, with each disk taking on a new letter. Partitions are numbered starting with 1, so you might refer to `/dev/sda2`, `/dev/sdb6`, and so on. When using MBR, partitions 1 through 4 are reserved for primary or extended partitions, whereas logical partitions take numbers 5 and up.

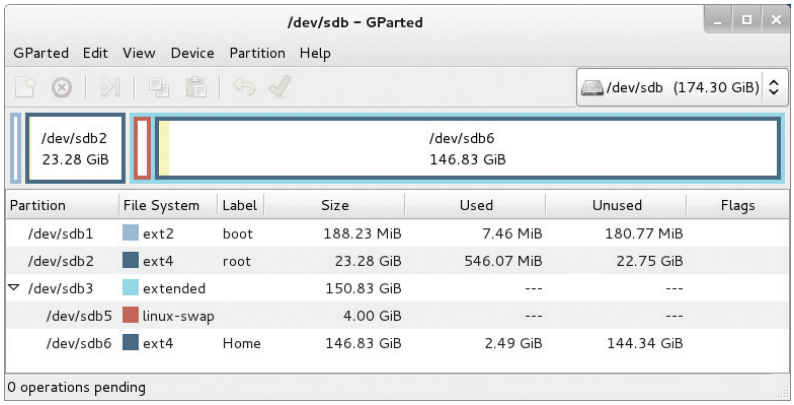

FIGURE 5.2 GParted, like other GUI disk partitioning tools, provides a graphical representation of your partitions.

Understanding Filesystem Issues

Most disk partitions contain filesystems, which are data structures that help the computer organize your directories and files. In Windows, each filesystem receives its own device letter, such as `A:` and `B:` for floppy disks, `C:` for the first hard disk partition (normally the boot partition), and so on. In Linux, by contrast, all filesystems are part of a single directory tree. The main filesystem is referred to as the root (`/`) filesystem. If a disk has multiple filesystem partitions, each is *mounted* at a *mount point* in the root (`/`) filesystem—that is, the contents of the additional filesystems are made available at specific directories, such as at `/home` (which holds users' data files) or `/boot` (which holds boot files). Several Linux filesystems exist, each with its own unique features:

Ext2fs The Second Extended Filesystem (ext2fs) was popular in the 1990s but is rarely used today because it lacks a *journal*, which is a filesystem feature that speeds filesystem checks after power outages or system crashes. A journal consumes disk space, though, so ext2fs is still useful on small disks. You might want to use it for a separate `/boot` partition, for instance, since such partitions are rather small. Its Linux filesystem type code is `ext2`.

The word *filesystem* is sometimes applied to the directory structure as a whole, even if it contains multiple low-level filesystems. Which meaning is intended is usually clear from the context.

The original Extended Filesystem (extfs) was used in the early 1990s but was quickly eclipsed by ext2fs. Extfs is no longer supported.

Ext3fs The Third Extended Filesystem (ext3fs) is essentially ext2fs with a journal. Until around 2010, it was a very popular filesystem, but ext4fs has taken its place. It supports files of up to 2 TiB and filesystems of up to 16 TiB (ext2fs imposes the same limits). Its Linux type code is `ext3`.

Ext4fs The Fourth Extended Filesystem (ext4fs) is a further development of the ext filesystem line. It adds speed improvements and the ability to handle larger files and disks—files may be up to 16 TiB in size, and filesystems may be up to 1 EiB (2^{60} bytes). Linux utilities refer to it as `ext4`.

Current partitioning tools limit ext4 filesystem sizes to 16 TiB; however, this limit will be raised sooner or later.

ReiserFS This filesystem, referred to as `reiserfs` by Linux tools, is similar to ext3fs in features, with an 8 TiB file-size limit and 16 TiB filesystem-size limit. Its best feature is its capacity to make efficient use of disk space with small files—those with sizes measured in the low kibibyte range. ReiserFS development has slowed, but it remains usable.

JFS IBM developed its Journaled File System (JFS) for its AIX OS, and its code eventually worked its way into Linux. JFS supports maximum file and filesystem sizes of 4 PiB and 32 PiB, respectively (1 PiB is 1024 TiB). JFS is not as popular as many other Linux filesystems. Linux tools use `jfs` as its type code.

XFS Silicon Graphics developed the Extents File System (XFS; Linux type code `xfs`) for its IRIX OS and later donated its code to Linux. XFS supports files of up to 8 EiB and filesystems of up to 16 EiB, making it the choice for *very* big disk arrays. XFS works well with large multimedia and backup files.

Btrfs This new filesystem (pronounced "butter-eff-ess" or "better-eff-ess") is intended as the next-generation Linux filesystem. It supports files of up to 16 EiB and filesystems of the same size. It also provides a host of advanced features, such as the ability to combine multiple physical disks into a single filesystem. As of early 2012, Btrfs is still experimental, but it may provide the best overall feature mix once it's finished. Its Linux type code is `btrfs`.

It's possible to use several filesystems in a single Linux installation, to take advantage of the benefits of each filesystem for different sets of files.

If you're planning a new Linux installation, you should consider ext4fs, ReiserFS, or XFS as your filesystems. Currently, ext4fs provides the best overall features and performance, while ReiserFS and XFS are worth considering for volumes that will hold particularly small and large files, respectively. Ext4fs is a good choice for volumes that hold large files, though, so you could use ext4fs for everything and not go far wrong, particularly on a general-purpose computer.

In addition to Linux's native filesystems, the OS supports several other filesystems, some of which are important in certain situations:

FAT The File Allocation Table (FAT) filesystem was the standard with DOS and Windows through Windows Me. Just about all OSs support it. Its compatibility also makes it a good choice for exchanging data between two OSs that dual-boot on a single computer. Unlike most filesystems, FAT has two Linux type codes: `msdos` and `vfat`. Using `msdos` causes Linux to use the filesystem as DOS did, with short filenames with at most 8 characters plus a 3-character extension (*8.3 filenames*); when you use `vfat`, Linux supports long filenames on FAT.

FAT's simplicity and widespread support make it a popular filesystem on floppy disks, USB flash drives, cell phones, e-book readers, digital camera media, and so on.

NTFS Microsoft developed the New Technology File System (NTFS) for Windows NT, and it is the default filesystem for recent versions of Windows. Linux provides a limited read/write NTFS driver, and a full read/write driver is available in the NTFS-3g software (`http://www.tuxera.com`). You're most likely to encounter it on a Windows boot partition in a dual-boot configuration or on larger removable or external hard disks. Under Linux, the standard kernel driver is known as `ntfs`, whereas the NTFS-3g driver is called `ntfs-3g`.

Linux's `ntfs` driver is based in the kernel, which makes it fast. The `ntfs-3g` driver, unlike most filesystem drivers, is *not* kernel-based, so it's not as fast.

HFS Apple used its Hierarchical File System (HFS) in Mac OS through version 9.*x* and still supports it in Mac OS X. You might encounter HFS on some removable media, and particularly on older disks created under pre-X versions of Mac OS. Linux provides full read/write HFS support using its `hfs` driver.

HFS+ Apple's HFS+, also known as *Mac OS Extended*, is the current filesystem for Mac OS X; you're likely to encounter it on dual-boot Macs and on some removable media created for use with Macs. Linux provides read/write HFS+ support with its `hfsplus` driver; however, write support is disabled by default on versions of the filesystem that include a journal.

Mac users often use FAT on removable media for compatibility reasons.

ISO-9660 This filesystem is used on optical media, and particularly on CD-ROMs and CD-Rs. It comes in several levels with differing capabilities. Two extensions, *Joliet* and *Rock Ridge*, provide support for long filenames using Windows and Unix standards, respectively. Linux supports all these variants. You should use the `iso9660` type code to mount an ISO-9660 disc.

UDF The Universal Disk Format (UDF) is a filesystem that's intended to replace ISO-9660. It's most commonly found on DVD and Blu-ray media, although it's sometimes used on CD-Rs as well. Its Linux type code is, naturally, `udf`.

Most non-Linux filesystems lack support for the Unix-style ownership and permissions that Linux uses. Thus, you may need to use special mount options to set ownership and permissions as you want them. Exceptions to this rule include HFS+ and ISO-9660 when Rock Ridge extensions are in use. Rock Ridge discs are generally created with ownership and permissions that enable normal use of the disc, but if you're faced with an HFS+ disk, you may find that the user ID (UID) values don't match those of your Linux users. Thus, you may need to copy data as `root`, create an account with a matching UID value, or change the ownership of files on the HFS+ disk. (This last option is likely to be undesirable if you plan to use the disk again under Mac OS X.)

To access a filesystem, you must mount it with the `mount` command. For instance, to mount the filesystem on `/dev/sda5` at `/shared`, you would type the following command:

```
# mount /dev/sda5 /shared
```

The `umount` command's name has just one `n`.

Alternatively, you can create an entry for the filesystem in the `/etc/fstab` file, which stores information such as the device file, filesystem type, and mount point. When you're done using a filesystem, you can unmount it with the `umount` command, as in **`umount /shared`**.

Using Removable and Optical Disks

Certification Objective

If you insert a removable disk into a computer that's running most modern Linux distributions, the computer will probably detect that fact, mount the disk in a subdirectory of `/media`, and launch a file manager on the disk. This behavior makes the system work in a way that's familiar to users of Windows or Mac OS.

Some devices, such as optical disc drives, can lock their eject mechanisms to prevent forced removal of the media.

When you're done using the disk, you *must* unmount it before you can safely remove it. Most file managers enable you to do this by right-clicking the entry for the disk in the left-hand pane and selecting an option called Unmount, Eject Volume, or Safely Remove, as shown in Figure 5.3. If you fail to do this, the filesystem may suffer damage.

Most removable disks are either unpartitioned or have a single partition. They frequently use FAT, which is a good choice for cross-platform compatibility. If you need to, you can partition USB flash drives and most other removable media.

Optical discs are unusual in that they require their own special filesystems (ISO-9660 or UDF). Although you can mount and unmount these discs just like other disks, you can only read them, not write to them. If you want to create an optical disc on blank media, you must use special software, such as the text-mode `mkisofs`, `cdrecord`, or `growisofs`, or the GUI K3B or X-CD-Roast. These tools create an ISO-9660 filesystem from the files you specify and then burn it to the blank disc. Thereafter, you can mount the disc in the usual way.

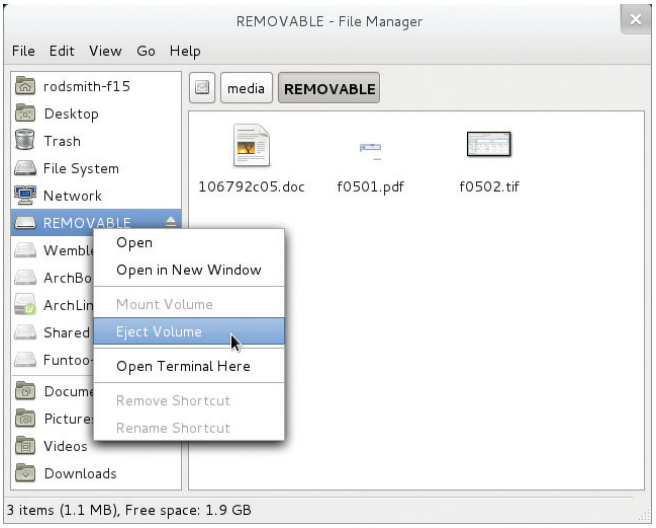

FIGURE 5.3 **Linux file managers enable you to unmount removable media.**

Managing Displays

Certification Objective

Linux provides two display modes: text-mode and GUI. A text-mode display is fairly straightforward and requires little or no management. GUI displays, on the other hand, are more complex. In Linux, the X Window System (or X for short) manages the GUI display. In the next few pages, I describe what X is and how X interacts with common display hardware.

Understanding the Role of X

Most people don't give much thought to the software behind their computers' displays; it all just works. Of course, behind the scenes the task of managing the display is fairly complex. Just some of the things that the software must do, on any platform, includes:

1. Initialize the video card, including set its resolution
2. Allocate sections of the display to hold windows that belong to particular applications
3. Manage windows that overlap so that only the "topmost" window's contents are displayed

4. Manage a pointer that the user controls via a mouse or similar device
5. Direct user input from a keyboard to whatever application is active
6. Display text and simple shapes within windows, at the request of user programs
7. Provide user interface elements to move and resize windows
8. Manage the interiors of windows, such as displaying menus and scroll bars

Desktop environments include window managers, but window managers without desktop environments are also available.

In Linux, tasks 1 through 6 are handled by X, but task 7 is handled by a program called a *window manager* and task 8 is handled by GUI libraries known as *widget sets*. The font display element of task 6 can be handled by X, but in recent years many programs have begun using a library called Xft for this task. Thus, the overall job of handling the display is broken up across several programs, although X handles most of the low-level tasks.

Modern Linux distributions use a version of X that can detect your hardware—including the video card, monitor, keyboard, and mouse—and configure itself automatically. The result is that the software normally works properly without any explicit configuration. Sometimes, though, this auto-configuration fails. When this happens, you must manually edit the X configuration file, `/etc/X11/xorg.conf`. If this file is missing, you can generate a sample file by typing the following command (with X *not* running) as `root`:

Chapter 11, "Editing Files," describes the `pico`, `nano`, and Vi text-mode text editors.

```
# Xorg -configure
```

The result is normally a file called `/root/xorg.conf.new`. You can copy this file to `/etc/X11/xorg.conf` and begin editing it to suit your needs. This task is complex and is beyond the scope of this book, but knowing the name of the file can help you get started—you can examine the file and locate additional documentation by searching on that name.

Using Common Display Hardware

Certification Objective

Much of the challenge in dealing with video devices is in managing drivers for the video chipsets involved. Most modern computers use one of a handful of video drivers:

- `nv`, `nouveau`, and `nvidia` work with NVIDIA video hardware.
- `ati` and `fglrx` work with AMD/ATI video hardware.
- `intel` works with Intel video hardware.

- The `fbdev` and `vesa` drivers are generic drivers that work with a wide variety of hardware, but they produce suboptimal performance.
- Many older video cards use more obscure drivers.

The `nvidia` and `fglrx` drivers are proprietary drivers provided by their manufacturers. Check the manufacturers' Web sites for details, or look for packages that install these drivers. These proprietary drivers provide features that aren't available in their open source counterparts, although the `nouveau` driver implements some of these features. In this context, video driver "features" translate into improved performance, particularly with respect to 3D graphics and real-time displays (as in playing back video files).

In the past, most video cards connected to monitors using a 15-pin Video Graphics Array (VGA) cable. Today, 29-pin Digital Visual Interface (DVI) cables are quite common. (Figure 5.4 shows both types.) DVI has the advantage of being a digital interface, which can produce a cleaner display on modern liquid crystal display (LCD) monitors.

High Definition Multimedia Interface (HDMI) is another cable type. HDMI and DVI are similar, but HDMI is more common on televisions, whereas DVI is more common on computers.

FIGURE 5.4 VGA connectors (left) were common in years past and are still available today; DVI connectors (right) are common on newer monitors and video cards.

Monitor resolutions are typically measured in terms of horizontal and vertical number of pixels. In the past, resolutions of as low as 640×480 have been common, but today it's rare to use a monitor that has an optimum resolution of lower than 1280×1024 or 1366×768, and resolutions of 1920×1080 or higher are commonplace. You should consult your monitor's manual to determine its optimum resolution. Typically, physically larger monitors have higher resolutions; however, this isn't always true.

Most monitors use an *aspect ratio* of either 4:3 or 16:9, referring to the ratio of the length to the height of the display.

In the best of all possible worlds, Linux will auto-detect your monitor's optimum resolution and set itself to that value whenever you boot the computer. Unfortunately, this sometimes doesn't work. Keyboard/video/mouse (KVM) switch boxes and extension cables can sometimes interfere with this auto-detection, and

older monitors might not support the necessary computer/monitor communication. Thus, you may need to crack open your monitor's manual to learn what its optimum resolution is. Look for this information in the features or specifications section; it will probably be called *optimum resolution*, *maximum resolution*, or something similar. It may also include a refresh rate value, as in *1280×1024 @ 60 Hz* .

In most cases, you can set the resolution using a GUI tool such as the Displays item in GNOME System Settings panel, shown in Figure 5.5. In Figure 5.5, the Resolution button enables you to set the resolution to any desired value. If you can't find the optimum resolution in the list, you may need to perform more advanced adjustments involving the `/etc/X11/xorg.conf` file—a topic that's beyond the scope of this book. On rare occasion, you may need to upgrade your video card; some cards aren't able to handle the optimum resolutions used by some monitors.

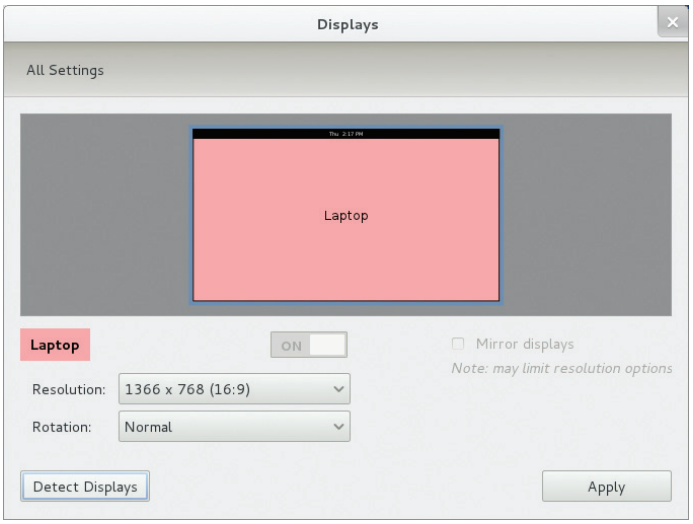

FIGURE 5.5 **Most desktop environments provide GUI tools to help you set your display's resolution.**

Handling USB Devices

Certification Objective

Most modern computers use USB as the primary interface for external peripherals. Keyboards, mice, cameras, flash storage, hard disks, network adapters, scanners, printers, and more can all connect via USB. For the most part, USB devices

work in a plug-and-play manner—you plug them in and they work. Some specific caveats include the following:

Human interface devices X usually takes over keyboards, mice, tablets, and similar devices when you plug them in. If you have problems, you may need to adjust your X configuration by editing `/etc/X11/xorg.conf`, but this is an advanced topic that's beyond the scope of this book.

Disk storage I include USB flash drives, external hard disks, and other storage devices in this category. As described earlier, in "Using Removable and Optical Disks," it's critical that you unmount the disk before you unplug its USB cable. Failure to do so can result in data corruption.

Cell phones, cameras, e-book readers, and music players You can often use these devices like disk devices to transfer photos, music, or other files. You may need to activate an option on the device to make it look like a disk device to the computer, though. When you're done, unmount the device and deactivate its interface mode.

Scanners Linux uses the Scanner Access Now Easy (SANE; `http://www.sane-project.org`) software to handle most scanners. A few require obscure or proprietary software, though.

Printers Most distributions automatically configure suitable printer queues when you plug in a USB printer. If you need to tweak the configuration, try entering **`http://localhost:631`** in a Web browser on the computer in question. Doing so opens a Web-based printer configuration utility. Some distributions provide distribution-specific printer configuration tools, as well.

Managing Drivers

Certification Objective

Most hardware devices require the presence of special software components to be useful. A piece of software that "talks" to hardware is known as a *driver*, so you should know how drivers work in Linux. Several broad classes of drivers exist, so I begin by describing those. Whatever the driver type, you should know how to locate and install drivers for your hardware.

Understanding Types of Drivers

Linux requires drivers because different hardware—even two devices that serve very similar purposes, such as two network adapters—can function in very different

In fact, several layers exist between the network hardware and a program like Firefox; the driver is just one of these layers.

ways. That is, the methods required to initialize and use two network adapters may be entirely different. To provide generalized interfaces so that programs like the Firefox Web browser can use any network adapter, the Linux kernel uses drivers as a bridge between the hardware-agnostic kernel interfaces and the hardware itself.

Broadly speaking, drivers can exist in one of two locations:

- The kernel
- A library or application

Most drivers are kernel based, and in fact a large chunk of the Linux kernel consists of drivers. Kernel drivers handle most of the devices that are internal to the computer's box, such as the hard disks, network interfaces, and USB interfaces. The kernel hosts most drivers because drivers typically require privileged access to hardware, and that's the purpose of the kernel.

Some drivers reside in a library or application. Many of these devices are external to the computer itself. Examples include:

- SANE, which handles scanners
- Ghostscript, which converts printed output into a form that particular printers can understand
- X, which manages the display

X is unusual in that it's a non-kernel element that communicates more or less directly with the video hardware. SANE and Ghostscript, by contrast, both communicate with external hardware devices via interfaces (such as a USB port) that are handled by the kernel. That is, you need at least two drivers to handle such devices. To print to a USB printer, for instance, you use the kernel's USB driver and a Ghostscript printer driver. Ideally, most users will be unaware of this complexity, but you may need to understand it if problems arise.

Locating and Installing Drivers

Most drivers come with the Linux kernel itself, or with the library or application that handles the type of hardware. For instance, most X installations include a set of video drivers so that you can use most video cards. For this reason, it's seldom necessary to track down and install additional drivers for common hardware. There are exceptions; for instance:

New hardware If your hardware is very new (meaning the model is new, not just that you purchased it recently), it might need drivers that haven't yet made their way into whatever distribution you're using.

Unusual hardware If you're using very exotic hardware, such as a specialized scientific data-acquisition board, you may need to track down drivers for it.

Proprietary drivers Some manufacturers provide proprietary drivers for their hardware. For instance, the `nvidia` and `fglrx` video drivers (referred to earlier, in "Using Common Display Hardware") can improve the performance of video displays based on NVIDIA or ATI/AMD chipsets, respectively. Some hardware requires proprietary drivers. This is particularly common for some exotic hardware.

Bug fixes Drivers, like other software, can be buggy. If you run into such a problem, you might want to track down a more recent driver to obtain a bug fix.

One way to obtain a new kernel-based driver is to upgrade the kernel. Note that a kernel upgrade can provide both bug fixes to existing drivers and entirely new drivers. Similarly, upgrading software such as SANE, Ghostscript, or X can upgrade or add new drivers for the devices that such packages handle.

If you're using exotic hardware or need some other hard-to-find driver, your task can be more difficult. You can check with the manufacturer or perform a Web search to try to find drivers.

If you obtain a driver that's not part of the kernel (or software package to handle the device), you should read the instructions that come with the driver. Installation procedures vary quite a bit from one driver to another, so it's impossible to provide a simple step-by-step installation procedure that works in all cases. Some drivers come with installation utilities, but others require you to follow a procedure that involves typing assorted commands. If you're very unlucky, the driver will come in the form of a *kernel patch*. This is a way to add or change files in the main kernel source code package. You must then recompile the kernel—a task that's well beyond the scope of this book.

The Essentials and Beyond

Software and hardware interact in numerous ways to determine a computer's capabilities. Your CPU is one determinant of your computer's speed, and it also determines what version of Linux you can run. CPUs are mounted on motherboards, which contain other critical circuitry for managing hard disks, displays, and other devices. Your hard disk must be partitioned and prepared with one or more filesystems before it's useful. Video hardware—both the monitor and the video circuitry inside the computer—determine how your desktop environment looks and how fast the computer can move windows and display videos. USB manages most external devices, such as keyboards, mice, and external disks. Software known as drivers manages all these hardware devices.

(Continues)

The Essentials and Beyond *(Continued)*

Suggested Exercises

- At a Linux shell prompt, type **`uname -a`**, **`lscpu`** , and cat **`/proc/cpuinfo`**. Compare the output and try to determine your CPU's capabilities. In particular, can it run 64-bit applications, and is your current distribution a 32-bit or 64-bit distribution?
- After you've logged into your preferred desktop environment, insert an optical disc, a USB flash drive, or some other removable disk. Does a file browser open up? If not, open one manually and try to find your removable disk. Once you've accessed the disk, unmount it so that you can safely remove it.

Review Questions

1. Which of the following commands provides the most information about your motherboard's features?

   A. `lscpu`
   B. `Xorg -configure`
   C. `fdisk -l /dev/sda`
   D. `lspci`
   E. `http://localhost:631`

2. Why might you want to partition a hard disk? (Select all that apply.)

   A. To install more than one OS on the disk
   B. To use ext4fs rather than ReiserFS
   C. To turn a PATA disk into an SATA disk
   D. To separate filesystem data from swap space
   E. To separate the disk's cache from its main data

3. Which of the following devices is *not* commonly attached via USB?

   A. Video monitors
   B. Keyboards
   C. External hard disks
   D. Printers
   E. Scanners

4. True or false: An EM64T CPU is capable of running a Linux distribution identified as being for the AMD64 CPU.
5. True or false: UDF is a good filesystem to use for a Linux installation on a hard disk.
6. True or false: The Linux kernel includes drivers for various disk controllers, network adapters, and USB interfaces, among other things.

(Continues)

The Essentials and Beyond *(Continued)*

7. The *x*86 CPU uses a ___-bit architecture.
8. A computer power supply converts electricity from alternating current to _____. (Two words)
9. The _____ standard is a modern video interface that's commonly used on computer monitors.