Chapter 1

Understanding the Basics

The Linux operating system has taken the world by storm. Whether it's embedded Linux software operating in phones and refrigerators or full-blown Linux servers running famous Internet sites, you can find Linux systems just about everywhere. If you've chosen (or have been chosen) to be a Linux system administrator, the task before you can seem daunting at first. But don't panic—while complex, the Linux system is organized and structured. Just knowing the basics of how Linux works will go a long way in helping you with your goals of becoming a Linux system administrator. This chapter walks you through the basics of what Linux is and explains the different versions of Linux that are available.

IN THIS CHAPTER, YOU WILL LEARN TO

- List the components of a standard Linux system
- Explain how GNU utilities are used in Linux
- Describe the various Linux user interface environments
- Explain why there are different Linux distributions

What Is Linux?

If you've never worked with Linux, you may be confused as to why there are so many different versions of it available. You've probably heard various terms such as *distribution*, *LiveDVD*, and *GNU* when looking at Linux packages, and may have been confused. Wading through the world of Linux for the first time can be a tricky experience. This chapter takes some of the mystery out of the Linux system before you start working on commands and scripts.

For starters, these four main parts make up a Linux system:

- The Linux kernel
- The GNU utilities
- A user interface
- Application software

Each of these four parts has a specific job in the Linux system. Figure 1.1 shows a basic diagram of how the parts fit together on top of the computer hardware to create the overall Linux system.

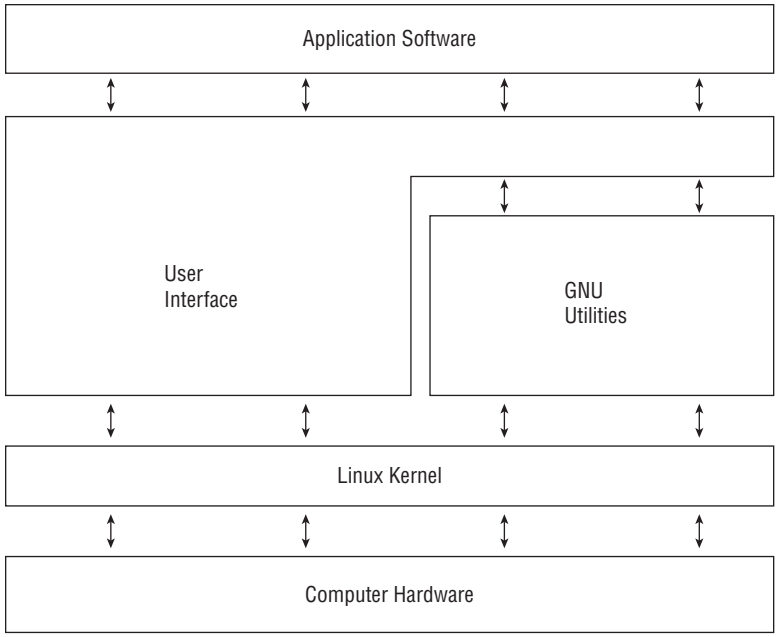

FIGURE 1.1
The Linux system

This section describes these four main parts in detail and gives you an overview of how they work together to create a complete Linux system.

Looking into the Linux Kernel

The core of the Linux system is the *kernel*. The kernel controls all of the hardware and software on the computer system, allocating hardware when necessary and executing software when required.

If you've been following the Linux world at all, no doubt you've heard the name Linus Torvalds. Linus is the person responsible for creating the first Linux kernel software while he was a student at the University of Helsinki. He intended it to be a copy of the Unix system, at the time a popular operating system used at many universities.

After developing the Linux kernel, Linus released it to the Internet community and solicited suggestions for improving it. This simple process started a revolution in the world of computer operating systems. Soon Linus was receiving suggestions from students as well as professional programmers from around the world.

Allowing anyone to change programming code in the kernel would result in complete chaos. To simplify things, Linus acted as a central point for all improvement suggestions. It was ultimately Linus's decision whether to incorporate suggested code in the kernel. This same concept is still in place with the Linux kernel code, except that instead of just Linus controlling the kernel code, a team of developers has taken on the task.

The kernel is primarily responsible for these four main functions:

- System memory management
- Software program management

- Hardware management
- Filesystem management

The following sections explore each of these functions in more detail.

System Memory Management

One of the primary functions of the operating system kernel is memory management. Not only does the kernel manage the physical memory available on the server, but it can also create and manage virtual memory, or memory that does not actually exist.

It does this by using space on the hard disk, called the *swap space*. The kernel swaps the contents of virtual memory locations back and forth from the swap space to the actual physical memory. This allows the system to think there is more memory available than what physically exists (shown in Figure 1.2).

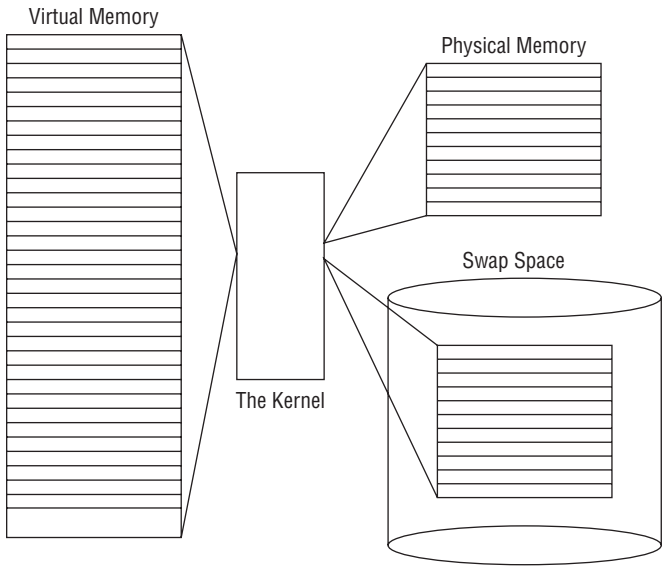

Figure 1.2
The Linux system memory map

The memory locations are grouped into blocks called *pages*. The kernel locates each page of memory either in the physical memory or in the swap space. The kernel then maintains a table of the memory pages that indicates which pages are in physical memory and which pages are swapped out to disk.

The kernel keeps track of which memory pages are in use and automatically copies memory pages that have not been accessed for a period of time to the swap space area (called *swapping out*), even if there's other memory available. When a program wants to access a memory page that has been swapped out, the kernel must make room for it in physical memory by swapping out a different memory page and swapping in the required page from the swap space. Obviously, this process takes time and can slow down a running process. The process of swapping out memory pages for running applications continues for as long as the Linux system is running.

Real World Scenario

Looking at Memory

There are a couple of simple commands you can use to get an idea of just how your Linux system is managing memory. While we'll be exploring these commands in more detail later in the book, here's a quick exercise for you to get started exploring your Linux system:

1. Log into your Linux system. (If you don't have a Linux system available yet, you can come back to here after going through either Chapter 2, "Installing an Ubuntu Server," or Chapter 4, "Installing a Red Hat Server.")
2. From the command prompt, enter the command **`free`**. You should see something similar to this output:

```
$ free
               total        used        free      shared  buff/cache   available
Mem:         2035504      135668     1449568        1048      450268     1742704
Swap:        2097148           0     2097148
```

   The output from the `free` command shows the total amount of physical memory installed on the system, as well as the amount of swap space currently configured.

3. The `free` command just provides an overview of the memory for your Linux system. For a more detailed look, enter the command **`cat /proc/meminfo`**. You should see a long listing, similar to this:

```
$ cat /proc/meminfo
MemTotal:        2035504 kB
MemFree:         1449632 kB
MemAvailable:    1742352 kB
Buffers:           25452 kB
Cached:           386028 kB
SwapCached:            0 kB
Active:           166036 kB
Inactive:         290704 kB
Active(anon):      51796 kB
Inactive(anon):      128 kB
Active(file):     114240 kB
Inactive(file):   290576 kB
Unevictable:       18640 kB
Mlocked:           18640 kB
SwapTotal:       2097148 kB
SwapFree:        2097148 kB
Dirty:               156 kB
Writeback:             0 kB
AnonPages:         63940 kB
Mapped:            63344 kB
Shmem:              1048 kB
```

```
KReclaimable:      38664 kB
Slab:              74316 kB
SReclaimable:      38664 kB
SUnreclaim:        35652 kB
KernelStack:        2044 kB
PageTables:         1268 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     3114900 kB
Committed_AS:     376812 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       27676 kB
VmallocChunk:          0 kB
Percpu:              516 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:       90048 kB
DirectMap2M:     2007040 kB
$
```

The kernel continually updates the `meminfo` file to show exactly what's going on in memory at that moment in time, so the file constantly changes.

Software Program Management

The Linux operating system calls a running program a *process*. A process can run in the foreground, displaying output on a display, or it can run in background, behind the scenes. The kernel controls how the Linux system manages all the processes running on the system.

The kernel creates the first process, called the *init process*, to start all other processes on the system. When the kernel starts, it loads the init process into virtual memory. As the kernel starts each additional process, it gives it a unique area in virtual memory to store the data and code that the process uses.

There are a few different types of init process implementations available in Linux, but these days, the two most popular are as follows:

SysVinit—The *SysVinit* (SysV) initialization method was the original method used by Linux and was based on the Unix System V initialization method. Though it is not used by many Linux distributions these days, you still may find it around in older Linux distributions.

Systemd—The *systemd* initialization method was created in 2010 and has become the most popular initialization and process management system used by Linux distributions.

The SysVinit initialization method used a concept called *runlevels* to determine what processes to start. The runlevel defines the state of the running Linux system and what processes should run in each state. Table 1.1 shows the different runlevels associated with the SysVinit initialization method.

TABLE 1.1: The SysVinit Runlevels

RUNLEVEL	DESCRIPTION
0	Shuts down the system
1	Single-user mode used for system maintenance
2	Multiuser mode without networking services enabled
3	Multiuser mode with networking services enabled
4	Custom
5	Multiuser mode with GUI available
6	Reboots the system

The `/etc/inittab` file defines the default runlevel for a system. The processes that start for specific runlevels are defined in subdirectories of the `/etc/rc.d` directory. You can view the current runlevel at any time using the `runlevel` command, as shown here:

```
$ runlevel
N 5
$
```

The systemd initialization method became popular because it has the ability to start processes based on different events such as these:

- When the system boots
- When a particular hardware device is connected
- When a service is started
- When a network connection is established
- When a timer has expired

The systemd method determines what processes to run by linking events to *unit files*. Each unit file defines the programs to start when the specified event occurs. The `systemctl` program allows you to start, stop, and list the unit files currently running on the system.

The systemd method groups unit files together into *targets*. A target defines a specific running state of the Linux system, similar to the SysVinit runlevel concept. At system startup, the *default.target* unit defines all the unit files to start. You can view the current default target using the `systemctl` command.

```
$ systemctl get-default
graphical.target
$
```

The `graphical.target` target defines the processes to start when a multiuser graphical environment is running, similar to the old SysVinit runlevel 5.

Real World Scenario

Examining Processes

In Chapter 14, "Working with Processes and Jobs," you'll see how to use the `ps` command to view the processes currently running on the Linux system. You can use it now to take a quick peek at what programs are currently running on your Linux system.

1. Log into your Linux system. (If you don't have a Linux system available yet, you can come back to here after going through either Chapter 2 or Chapter 4.)
2. At the command prompt, enter the command **`ps ax`**. You should see something similar to this output:

```
$ ps ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:00 /sbin/init maybe-ubiquity
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
      5 ?        I      0:00 [kworker/0:0-memcg_kmem_cache]
      6 ?        I<     0:00 [kworker/0:0H-kblockd]
      7 ?        I      0:00 [kworker/0:1-events]
      8 ?        I      0:00 [kworker/u2:0-events_power_efficient]
. . .
   1033 tty1     S      0:00 -bash
   1054 tty1     R+     0:00 ps ax
$
```

We've just shown the start of the listing, along with the last two lines, but you should see a long list of different programs running on your Linux system (including the `ps` command that you started). The kernel is keeping track of all those programs!

Hardware Management

Still another responsibility for the kernel is hardware management. Any device that the Linux system must communicate with needs driver code inserted inside the kernel code. The driver code allows the kernel to pass data back and forth to the device, acting as a middleman between applications and the hardware. There are two methods used for inserting device driver code in the Linux kernel.

- Drivers compiled in the kernel
- Driver modules added to the kernel

Previously, the only way to insert device driver code was to recompile the kernel. Each time you added a new device to the system, you had to recompile the kernel code. This process became even more inefficient as Linux kernels supported more hardware. Fortunately, Linux developers devised a better method to insert driver code into the running kernel.

Programmers developed the concept of kernel modules to allow you to insert driver code into a running kernel without having to recompile the kernel. Also, a kernel module could be removed from the kernel when the device was finished being used. This greatly simplified and expanded using hardware with Linux.

The Linux system identifies hardware devices as special files, called *device files*. There are three different classifications of device files.

- Character
- Block
- Network

Character device files are for devices that can only handle data one character at a time. Most types of modems and terminals are created as character files. Block files are for devices that can handle data in large blocks at a time, such as disk drives.

The network file types are used for devices that use packets to send and receive data. This includes network cards and a special loopback device that allows the Linux system to communicate with itself using common network programming protocols.

Linux creates special files, called *nodes*, for each device on the system. All communication with the device is performed through the device node. Each node has a unique number pair that identifies it to the Linux kernel. The number pair includes a major and a minor device number. Similar devices are grouped into the same major device number. The minor device number is used to identify a specific device within the major device group.

Filesystem Management

Unlike some other operating systems, the Linux kernel can support different types of filesystems to read and write data to and from hard drives. Besides having more than a dozen filesystems of its own, Linux can read and write to and from filesystems used by other operating systems, such as Microsoft Windows. The kernel must be compiled with support for all types of filesystems that the system will use. Table 1.2 lists the standard filesystems that a Linux system can use to read and write data.

Table 1.2: Linux Filesystems

Filesystem	Description
ext	Linux extended filesystem—the original Linux filesystem
ext2	Second extended filesystem; provides advanced features over ext
ext3	Third extended filesystem; supports journaling
ext4	Fourth extended filesystem; supports advanced journaling
btrfs	A newer, high-performance filesystem that supports journaling and large files
exfat	The extended Windows filesystem, used mainly for SD cards and USB sticks
hpfs	OS/2 high-performance filesystem
jfs	IBM's journaling file system
iso9660	ISO 9660 filesystem (CD-ROMs)
minix	MINIX filesystem
msdos	Microsoft FAT16
ncp	NetWare filesystem
nfs	Network File System
ntfs	Support for Microsoft NT filesystem
proc	Access to system information
smb	Samba SMB filesystem for network access
sysv	Older Unix filesystem
ufs	BSD filesystem
umsdos	Unix-like filesystem that resides on top of msdos
vfat	Windows 95 filesystem (FAT32)
XFS	High-performance 64-bit journaling filesystem

Any hard drive that a Linux server accesses must be formatted using one of the filesystem types listed in Table 1.2.

The Linux kernel interfaces with each filesystem using the Virtual File System (VFS). This provides a standard interface for the kernel to communicate with any type of filesystem. VFS caches information in memory as each filesystem is mounted and used.

The GNU Utilities

Besides having a kernel to control hardware devices and launch programs, a computer operating system needs utilities to perform standard functions, such as controlling files and programs. While Linus created the Linux system kernel, he had no system utilities to run on it. Fortunately for him, at the same time he was working, a group of people were working together on the Internet trying to develop a standard set of computer system utilities that mimicked the popular Unix operating system.

The GNU organization (GNU stands for GNU's Not Unix) developed a complete set of Unix utilities but had no kernel system to run them on. These utilities were developed under a software philosophy called open source software (OSS).

The concept of OSS allows programmers to develop software and then release it to the world with no licensing fees attached. Anyone can use the software, modify it, or incorporate it into their own system without having to pay a license fee. Uniting Linus's Linux kernel with the GNU operating system utilities created a complete, functional, free operating system.

While the bundling of the Linux kernel and GNU utilities is often just called Linux, you will see some Linux purists on the Internet refer to it as the GNU/Linux system to give credit to the GNU organization for its contributions to the cause.

The GNU project was mainly designed for Unix system administrators to have a Unix-like environment available. This focus resulted in the project porting many common Unix system command-line utilities. The core bundle of utilities supplied for Linux systems is called the *coreutils* package.

The GNU coreutils package consists of these three parts:

- Utilities for handling files
- Utilities for manipulating text
- Utilities for managing processes

Each of these three main groups of utilities contains several utility programs that are invaluable to the Linux system administrator and programmer.

Linux User Interfaces

Having a world-class operating system that can manage your computer hardware and software is great, but you also need some way to communicate with it. Back in the old days of computers, you communicated with the mainframe computer by punching holes into cards, feeding them into a card reader, and then waiting for the output to appear on a printer. Fortunately, those days are long gone.

Thanks to the Apple macOS and Microsoft Windows operating systems, these days most desktop computer users expect some type of graphical display to interact with their system. Linux doesn't disappoint, offering a plethora of graphical desktops you can choose from. The following sections describe a few of the more popular ones.

The X Window Software

Two basic elements control your video environment—the video card in your PC and your monitor. To display fancy graphics on your computer, the Linux software needs to know how to talk to both of them. The X Window software is the core element in presenting graphics.

The X Window software is a low-level program that works directly with the video card and monitor in the PC and controls how Linux applications can present fancy windows and graphics on your computer.

Linux isn't the only operating system that uses X Window; there are versions written for many different operating systems. In the Linux world, there are a few different software packages that can implement it. There are two X Window packages that are most commonly used in Linux:

- `X.org`
- Wayland

The `X.org` package is the older of the two, based on the original Unix X Window System version 11 (often called X11). More Linux distributions are migrating to the newer Wayland software, which is more secure and easier to maintain.

When you first install a Linux distribution, it attempts to detect your video card and monitor and then creates an X Window configuration file that contains the required information. During installation, you may notice a time when the installation program scans your monitor for supported video modes. Sometimes this causes your monitor to go blank for a few seconds. Because there are lots of different types of video cards and monitors out there, this process can take a little while to complete.

The core X Window software produces a graphical display environment, but nothing else. While this is fine for running individual applications, it is not too useful for day-to-day computer use. There is no desktop environment allowing users to manipulate files or launch programs. To do that, you need a desktop environment on top of the X Window system software.

The KDE Plasma Desktop

The K Desktop Environment (KDE) was first released in 1996 as an open source project to produce a graphical desktop similar to the Microsoft Windows environment. The KDE desktop incorporates all of the features you are probably familiar with if you are a Windows user. Figure 1.3 shows the current version, called KDE Plasma, running in the openSUSE Linux distribution.

Figure 1.3
The KDE Plasma desktop on an openSUSE Linux system

The KDE Plasma desktop allows you to place both application and file icons in a special area on the desktop. If you single-click an application icon, the Linux system starts the application. If you single-click a file icon, the KDE desktop attempts to determine what application to start to handle the file.

The bar at the bottom of the desktop is called the Panel. The Panel consists of these four parts:

The KDE Start menu—Much like the Windows Start menu, the KDE Start menu contains links to start installed applications.

Program shortcuts—These are quick links to start applications directly from the Panel.

The taskbar—The taskbar shows icons for applications currently running on the desktop.

Applets—These are small applications that have an icon in the Panel that often can change depending on information from the application.

All of the Panel features are similar to what you would find in Windows. In addition to the desktop features, the KDE project has produced a wide assortment of applications that run in the KDE environment.

The GNOME Desktop

The GNU Network Object Model Environment (GNOME) is another popular Linux desktop environment. First released in 1999, GNOME has become the default desktop environment for many Linux distributions (the most popular being Red Hat Linux).

GNOME Controversy

The GNOME desktop underwent a radical change with version 3, released in 2011. It departed from the standard look and feel of most desktops using standard menu bars and taskbars to make the interface more user-friendly across multiple platforms, such as tablets and mobile phones. This change led to controversy (see the "Other Desktops" section), but slowly many Linux enthusiasts accepted the new look and feel of the GNOME 3 desktop.

Figure 1.4 shows the standard GNOME desktop used in the Ubuntu Linux distribution.

The GNOME 3 desktop cleans up the desktop interface by reducing the available menus to just these three:

Activities—Displays favorites, as well as any running application icons

Calendar—Shows the current date/time, along with any system notification messages

System—Shows network connections, system settings, and options to restart the system

The GNOME 3 desktop was designed to work on multiple types of devices, so you'll find there aren't a lot of menus. To launch applications, you must search for them using the Activities Overview, which is a search feature from the Activities menu.

Figure 1.4
A GNOME 3 desktop on an Ubuntu Linux system

Not to be outdone by KDE, the GNOME developers have also produced a host of graphical applications that integrate with the GNOME desktop.

Other Desktops

One of the main features of Linux is choice, and nowhere is that more evident than in the graphical desktop world. There are a plethora of different types of graphical desktops available in the Linux world. If you're not happy with the default desktop in your Linux distribution, it usually doesn't take much effort to change it to something else.

When the GNOME desktop project radically changed its interface in version 3, many Linux developers who preferred the look and feel of GNOME version 2 created spin-off versions based on GNOME 2. Of these, two became somewhat popular.

Cinnamon—The Cinnamon desktop was developed in 2011 by the Linux Mint distribution in an attempt to continue development of the original GNOME 2 desktop. It's now available as an option in several Linux distributions, including Ubuntu, Fedora, and openSUSE.

MATE—The MATE desktop was also developed in 2011 by an Arch Linux user who disliked the switch to GNOME 3. However, it incorporates a few features of GNOME 3 (such as replacing the taskbar) but maintains the overall look and feel of GNOME 2.

Figure 1.5 shows the Cinnamon desktop as it appears in the Linux Mint distribution.

The downside to these fancy graphical desktop environments is that they require a fair amount of system resources to operate properly. In the early days of Linux, a hallmark and selling feature of Linux was its ability to operate on older, less powerful PCs that the newer Microsoft desktop products couldn't run on. However, with the popularity of KDE Plasma and GNOME 3 desktops, this has changed, as it takes just as much memory to run a KDE Plasma or GNOME 3 desktop as the latest Microsoft desktop environment.

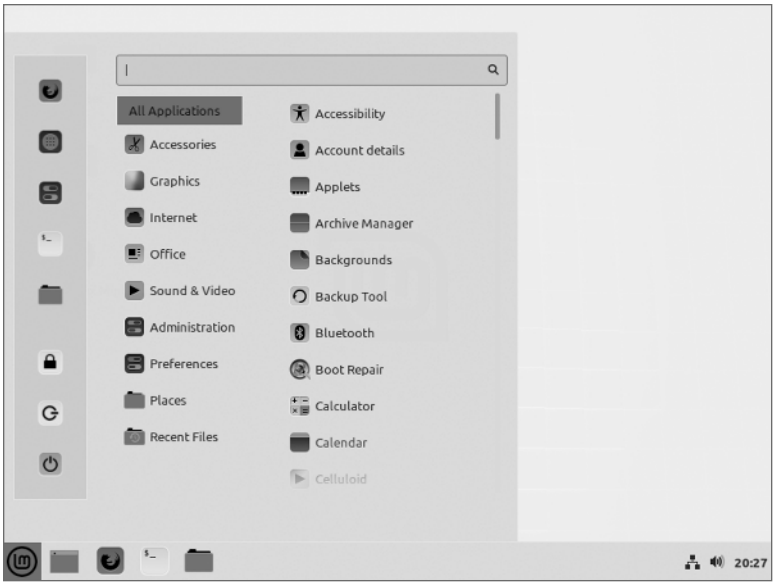

FIGURE 1.5
The Cinnamon desktop from Linux Mint

If you have an older PC, don't be discouraged. The Linux developers have banded together to take Linux back to its roots. They've created several low-memory–oriented graphical desktop applications that provide basic features that run perfectly fine on older PCs.

While these graphical desktops don't have a plethora of applications designed around them, they still run many basic graphical applications that support features such as word processing, spreadsheets, databases, drawing, and, of course, multimedia support.

Table 1.3 shows some of the smaller Linux graphical desktop environments that can be used on lower-powered PCs and laptops.

TABLE 1.3: Other Linux Graphical Desktops

DESKTOP	DESCRIPTION
Fluxbox	A bare-bones desktop that doesn't include a Panel, only a pop-up menu to launch applications
Xfce	A desktop that's similar to the GNOME 2 desktop, but with fewer graphics for low-memory environments
JWM	Joe's Window Manager, a lightweight desktop ideal for low-memory and low-disk space environments
fvwm	Supports some advanced desktop features such as virtual desktops and Panels, but runs in low-memory environments
fvwm95	Derived from fvwm, but made to look like a Windows 95 desktop

These graphical desktop environments are not as fancy as the KDE Plasma and GNOME 3 desktops, but they provide basic graphical functionality just fine. Figure 1.6 shows what the Xfce desktop used in the MX Linux distribution looks like.

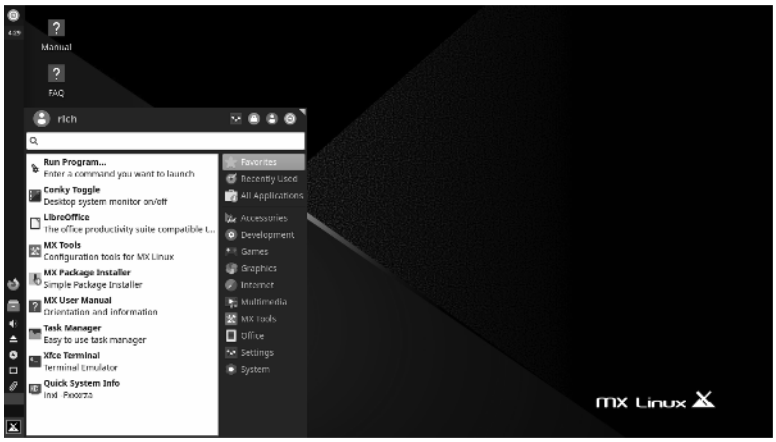

Figure 1.6
The Xfce desktop as seen in the MX Linux distribution

If you are using an older PC, try a Linux distribution that uses one of these desktops and see what happens. You may be pleasantly surprised.

The Command-Line Interface

While having a fancy graphical desktop interface is nice, there are drawbacks. The extra processing power required to interact with the graphics card takes away crucial CPU time that can be used for other programs. Nowhere is this more important than in a server environment.

Because of that, many Linux servers don't load a graphical desktop and instead rely on a text-based interface, called the *command-line interface* (CLI). The CLI provides a way for users to start programs, manage files on the filesystem, and manage processes running on the Linux system using simple text commands. The CLI is produced by a program called a *shell*. The shell allows you to enter text commands, and then it interprets the commands and then executes them in the kernel.

The shell contains a set of internal commands that you use to control things such as copying files, moving files, renaming files, displaying the programs currently running on the system, and stopping programs running on the system. Besides the internal commands, the shell also allows you to enter the name of a program at the command prompt. The shell passes the program name off to the kernel to start it.

You can also group shell commands into files to execute as a program. Those files are called *shell scripts*. Any command that you can execute from the command line can be placed in a shell script and run as a group of commands. This provides great flexibility in creating utilities for commonly run commands, or processes that require several commands grouped together.

There are quite a few Linux shells available to use on a Linux system. Different shells have different characteristics, some being more useful for creating scripts and some being more useful for managing processes. The default shell used in all Linux distributions is the Bash Shell. The Bash Shell was developed by the GNU project as a replacement for the standard Unix shell,

called the Bourne shell (after its creator). The Bash Shell name is a play on this wording, referred to as the "Bourne again shell."

In addition to the Bash Shell, there are several other popular shells you could run into in a Linux environment. Table 1.4 lists the more popular ones.

TABLE 1.4: Linux Shells

SHELL	DESCRIPTION
ash	A simple, lightweight shell that runs in low-memory environments but has full compatibility with the Bash Shell
korn	A programming shell compatible with the Bourne shell but supporting advanced programming features like associative arrays and floating-point arithmetic
tcsh	A shell that incorporates elements from the C programming language into shell scripts
zsh	An advanced shell that incorporates features from bash, tcsh, and korn, providing advanced programming features, shared history files, and themed prompts

Most Linux distributions include more than one shell, although usually they pick one of them to be the default. If your Linux distribution includes multiple shells, feel free to experiment with different shells and see which one fits your needs.

Linux Distributions

Now that you have seen the four main components required for a complete Linux system, you may be wondering how you are going to get them all put together to make a Linux system. Fortunately, there are people who have already done that for you.

A complete Linux system package is called a *distribution*. There are lots of different Linux distributions available to meet just about any computing requirement you could have. Most distributions are customized for a specific user group, such as business users, multimedia enthusiasts, software developers, or average home users. Each customized distribution includes the software packages required to support specialized functions, such as audio- and video-editing software for multimedia enthusiasts, or compilers and integrated development environments (IDEs) for software developers.

The different Linux distributions are often divided into two categories.

- Core Linux distributions
- Specialized distributions

The following sections describe these different types of Linux distributions and show some examples of Linux distributions in each category.

Core Linux Distributions

A core Linux distribution contains a kernel, one or more graphical desktop environments, and just about every Linux application that is available, precompiled for the kernel. It provides one-stop shopping for a complete Linux installation. Table 1.5 shows some of the more popular core Linux distributions.

Table 1.5: Core Linux Distributions

Distribution	Description
Slackware	One of the original Linux distribution sets, popular with Linux geeks
Red Hat Enterprise	A commercial business distribution used mainly for Internet servers
Gentoo	A distribution designed for advanced Linux users, containing only Linux source code
openSUSE	Different distributions for business and home use
Debian	Popular with Linux experts and commercial Linux products

In the early days of Linux, a distribution was released as a set of floppy disks. You had to download groups of files and then copy them onto disks. It would usually take 20 or more disks to make an entire distribution! Needless to say, this was a painful experience.

Nowadays, Linux distributions are released as an *ISO image file*. The ISO image file is a complete disk image of a DVD as a single file. You use a software application to either burn the ISO image file onto a DVD or create a bootable USB stick. You then just boot your workstation from the DVD or USB stick to install Linux. This makes installing Linux much easier.

However, beginners still often run into problems when they install one of the core Linux distributions. To cover just about any situation in which someone might want to use Linux, a single distribution has to include lots of application software. They include everything from high-end Internet database servers to common games.

While having lots of options available in a distribution is great for Linux geeks, it can become a nightmare for beginning Linux users. Most core distributions ask a series of questions during the installation process to determine which applications to load by default, what hardware is connected to the PC, and how to configure the hardware. Beginners often find these questions confusing. As a result, they often either load way too many programs on their computer or don't load enough and later discover that their computer won't do what they want it to do.

Fortunately for beginners, there's a much simpler way to install Linux.

Specialized Linux Distributions

A new subgroup of Linux distributions has started to appear. These are typically based on one of the main distributions but contain only a subset of applications that would make sense for a specific area of use.

In addition to providing specialized software (such as only office products for business users), customized Linux distributions also attempt to help beginning Linux users by autodetecting and

autoconfiguring common hardware devices. This makes installing Linux a much more enjoyable process.

Table 1.6 shows some of the specialized Linux distributions available and what they specialize in.

TABLE 1.6: Specialized Linux Distributions

DISTRIBUTION	DESCRIPTION
Fedora	A free distribution, originally used as a testing ground for Red Hat Enterprise Linux, now a popular distribution in its own right
Ubuntu	A free distribution originally intended for school and home desktop use, now also used as a server
MX Linux	A free distribution for home use
Linux Mint	A free distribution for home entertainment use
Puppy Linux	A free small distribution that runs well on older PCs

That's just a small sampling of specialized Linux distributions. There are hundreds of specialized Linux distributions, and more are popping up all the time on the Internet. No matter what your specialty, you'll probably find a Linux distribution made for you.

Many of the specialized Linux distributions are based on the Debian Linux distribution. They use the same installation files as Debian but package only a small fraction of a full-blown Debian system.

THE LINUX LiveDVD

Most Linux distributions also have a LiveDVD version available. The LiveDVD version is a self-contained ISO image file that you can burn onto a DVD (or USB stick) to boot up a running Linux system directly, without having to install it on your hard drive. Depending on the distribution, the LiveDVD either contains a small subset of applications or, in the case of specialized distributions, the entire system. The benefit of the LiveDVD is that you can test it with your system hardware before going to the trouble of installing the system.

The Bottom Line

List the components of a standard Linux system. The main components of a Linux system include the Linux kernel, the GNU utilities, a user interface, and application programs. The kernel controls how memory, programs, and hardware all interact with one another. The GNU

utilities provide useful functions such as text and file manipulation. The Linux user interfaces range from fancy graphical desktops, such as GNOME or KDE Plasma, to simple command-line interfaces, such as the Bash Shell.

> **Master It** The Linux kernel is constantly updated and managed by a group of developers. They publish their work at the `kernel.org` website. Go to that website and determine the version number of the latest stable release. What version is currently under development?

Explain how GNU utilities are used within Linux. The GNU utilities provide command-line functions for creating, modifying, moving, and deleting files, as well as working with data inside text files. The main GNU utilities are in the coreutils package.

> **Master It** The GNU community is constantly making improvements to the core GNU utilities used in Linux. You can find the latest released utilities at `www.gnu.org/software/coreutils/`. Go to that website and determine the current version of the GNU coreutils package.

Describe the various Linux user interface environments. There are many graphical desktop environments available in Linux. The two most popular ones are GNOME and KDE Plasma. Both provide common desktop features most desktop users are comfortable with. However, for server environments, it's most common to use a command-line interface (CLI) provided by a Linux shell program. The most common Linux shell is bash.

> **Master It** The GNU Bash Shell is continually being updated, with updates available at `www.gnu.org/software/bash/`. According to that website, what is the most recent version of bash available for download?

Explain why there are different Linux distributions. A Linux distribution bundles the various parts of a Linux system into a simple package that you can easily install on your PC. The Linux distribution world consists of full-blown Linux distributions that include just about every application imaginable, as well as specialized Linux distributions that only include applications focused on a special function.

> **Master It** There are many websites that track Linux distributions. The `www.distrowatch.com` website is a popular place to get information on new releases for lots of different distributions. Go to that site and list the current top five Linux distribution downloads.