1 What is Unix, and Why Is It Good?

Mac OS X is the most significant advance in desktop computing since the introduction of the original Mac interface. It provides users with unparalleled stability, flexibility, and openness—arguably necessary additions to the Macintosh operating system. Each of these features will help users take their Macs to new creative heights. Mac OS X provides these features by virtue of having been built on top of Unix. So if you're using Mac OS X, you're using Unix.

You probably already know that Unix is an industrial-strength operating system. It's specifically designed for always-on, network-connected computers that run multiple applications and are shared by many users. Since its creation in 1969, Unix has evolved into one of the world's most popular operating systems. Moreover, Unix is an excellent environment for creating new software. Apple built Mac OS X on top of a version of Unix called Darwin. If you're a Mac user who wants to push the boundaries of what you can do with your computer, here's what Unix can do for you.

The Advantages of a Unix-based Mac OS

While Unix is best known as a server operating system—most of the servers on the Internet run Unix—it's also been the desktop operating system for engineers, software developers, and system administrators. But Mac OS X is placing a Unix-based system on millions of desktops. With Unix under the hood of OS X, Macintosh users will now be able to take advantage of software developments beyond the boundaries of Apple Computer, while still enjoying the elegance and ease of use the Mac OS is famous for.

Basing Mac OS X on the Darwin operating system gave it three important features that, for all its advantages, the Mac OS had not previously had: stability, flexibility, and openness.

Stability

Even the most devoted Macintosh user will admit that system crashes have been an unfortunate but predictable part of everyday life. Unix systems, however, are extremely difficult to crash. Thanks to *protected memory*—the memory each application uses that is unavailable to any other application—with OS X your Macintosh will continue running even when one or more applications crash. You can simply restart the crashed application without having to restart your Mac.

If your system doesn't have protected memory, a badly behaving application can disturb the memory space of another application, or even of the operating system itself—often with nasty results. Macintosh operating systems before OS X didn't include protected memory—which explains all those system crashes!

A feature called *preemptive multitasking* allows the operating system to limit the amount of computational resources devoted to each application by prioritizing between tasks. Before Mac OS X, the Mac OS employed *cooperative* multitasking—in which each application is *supposed* to behave and play well with the other applications on a machine. You can guess what happens when cooperatively multitasked applications don't cooperate.

Flexibility

Unix was designed to allow different programs to be connected in an almost infinite variety of ways. Because thousands of utilities are available for Unix (and because they work together so well), Unix users can customize their work environments relatively easily, building their own tools when the need arises. Mac OS X itself comes with around 500 utilities, most of which can be easily combined with other programs (see Chapter 4, "Useful Unix Utilities," for a roundup of the ones you're most likely to use). Because of this (and its portability), Unix is the ideal environment for developing new software.

Openness

Darwin, like other open-source versions of Unix, such as Linux, is open—that is, the inner workings are open to examination and change. You can download, study, and alter its programming source code at will. (In fact, versions of Darwin other than Apple's already exist.) Say you want to create a server that enables you to synchronize an iPod with

What's in a Name?

Strictly speaking, Unix is a trademarked term that's been variously owned by AT&T, Novell, and now the Open Group (www.opengroup.org). Only Unix versions with the correct legal pedigree can use that name. In reality, though, most people casually refer to all of the various "flavors" as Unix.

any computer over the Internet, or one that sends faxes on demand from a catalog of files: Whatever software you create is likely to use an existing piece of Unix software as its starting point.

By allowing people to examine and change their operating systems, open-source software is central to the ongoing evolution and spread of Unix, resulting in a software-development environment that will continue to increase in stability, flexibility, and power.

Unix's ability to connect different programs together provides almost infinite flexibility. Combine this with Unix's built-in support for TCP/IP (the networking protocol that defines the Internet) and other networking tools, and you have an operating system—Mac OS X—that's ready to take you into a future in which you can build your applications, and every computer has the ability to be a server.

Whatever capability you want to add to an application, you can probably do it in Darwin. As a Mac enthusiast, you may have had limited exposure to any kind of programming, but you'll be able to expand your horizons by accessing the Unix underlying Mac OS X. Even if you're brand-new to programming, delving into Unix is the best way to start.

From Multics to Unix

Unix wasn't actually named until about a year into its development—at which point the wordplay on the preceding Multics project was intentional (*uni*, meaning "one," as opposed to *multi*, meaning "many"). The tradition of puns and word games in Unix software continues to this day, as you'll see in later chapters when we introduce programs such as `less`, which is an improvement on an earlier program called `more`. More became less, you see.

Becoming a sophisticated user

Thus, you should read this book because you want a deeper understanding of your computer, and to get your fingers and hands and mind inside of it. Using Unix is about moving from being a consumer of software and systems to being a creator of software and systems. This means pushing the envelope of how you interact with the operating system, delving into areas where most users don't go, in order to develop capabilities that most users don't have. Mac OS X makes this possible now because Unix is an operating system for developing and building, for getting into the nitty-gritty.

Since much Unix software is created by volunteers, you're benefiting from the hard work of thousands of users**.** But using Unix means you will always tweak, modify, and configure to get the software to do what you want. By bringing an industrial-strength server to your desktop, Apple has taken desktop computing to another level—in much the same way that Macintosh-plus-PostScript laser printers brought high-quality print publishing and graphics tools to the desktop. None of this is automatic, though, and using Unix places a greater burden on you. If you're coming to Unix expecting the shrink-wrapped experience of the Mac OS or Windows, you're bound to be disappointed.

You will work with Unix primarily from the command line in the Terminal application (more about that in the next chapter). You will put together lots of odd-sounding commands, creating tiny and not-so-tiny scripts and programs to give the machine capabilities it never had before. You will not only be customizing your machine and creating software, you will be customizing your world, and indirectly the world the rest of us live in.

Most people won't really notice that Mac OS X is Unix-based. In fact, most people will use their Mac OS X Macintoshes just as they always have—writing in their word processors, creating images in graphics software, and editing sound and video.

Furthermore, some of the Unix tools in Mac OS X were available in some form for Mac OS 9, but the Unix versions are included with Mac OS X (for example, a Web server and an email server). With Mac OS X, you are more likely to work with these applications, for a couple of reasons. Mac OS X is so stable that you won't be afraid of messing up your computer. More important, if you use a "pure" Unix tool, like the Apache Web server, then the skills you learn, and the system you build, will be transferable to almost any other Unix environment with little effort.

Unix will always be more hands-on than the graphical user interfaces you're accustomed to. However, if you're ready to experience new heights of computing creativity, you'll find that you have a more personalized and robust system on your hands by the time you finish this book.

Other Versions of Unix

Over the years, many versions of Unix have been developed—some by large corporations (for example, Sun Microsystems' Solaris and Hewlett-Packard's HP/UX), and some by small companies and individuals working for their own pleasure. The most famous of the latter is the open-source GNU/Linux operating system, which combines the work of hundreds of programmers from around the world and has been adopted by thousands of companies. (For example, IBM announced in January 2002 a mainframe computer designed specifically for Linux.)

A good list of dozens of versions of Unix is available at www.ugu.com/sui/ugu/show?ugu.flavors

But First, a Little History

What we wanted to preserve was not just a good environment in which to do programming, but a system around which a fellowship could form. We knew from experience that the essence of communal computing, as supplied by remote-access, time-shared machines, is not just to type programs into a terminal instead of a keypunch, but to encourage close communication.

—Dennis M. Ritchie, coinventor of Unix
From "The Evolution of the Unix Time-Sharing System," AT&T Bell Laboratories Technical Journal 63, No. 6, Part 2, October 1984 (http://cm.bell-labs.com/cm/cs/who/dmr/hist.html)

How did Unix end up as the underpinning of the Mac OS? In 1997, after a series of unsuccessful attempts to update the Macintosh operating system—remember Pink and Copland?—Apple bought NeXT, the computer company that Apple cofounder Steve Jobs had started 12 years earlier after he was forced out of Apple. NeXT had developed a powerful operating system with an elegant user interface but had failed to become commercially successful. (Of course, commercial success is not the only way to gauge the quality of a product. None other than Tim Berners-Lee, inventor of the World Wide Web, used a NeXT machine for his development work on hypertext.)

When Apple bought NeXT, it got the code for NeXT's operating system, development tools, and user interface. But more important, it got Steve Jobs and a culture of Unix-based development. The NeXT operating system, while largely written from scratch, was a version of Unix, and the NeXT engineers were used to a Unix culture—that is, employing powerful, flexible tools and systems in an environment of creative engineering. Given the effect on Apple's operating-system development, some people say that, culturally, NeXT bought Apple.

Since Mac OS X is based on a new version of Unix called Darwin, by the time OS X was released, the percentage of NeXT-derived code was small. Still, that cultural influence has played a huge part in moving Apple toward the values of openness, flexibility, and stability.

Where did those values come from? They were part of Unix from its beginnings. Unix was born in 1969 from the efforts of a small group of scientists working at AT&T's Bell Labs to create an operating system that would allow the group to continue the kind of collaborative programming they had been doing on an earlier project called Multics. Thus, from the very

continues on next page

Unix Pioneer: Bill Joy

Perhaps most recognizable to the general population as the chief scientist and cofounder of Sun Microsystems, Bill Joy is known in the Unix community as the primary designer of the Berkeley Software Distribution (BSD) version of Unix.

Among Joy's many contributions are the NFS (Network File System) protocol, the open-source version of TCP/IP, and the `vi` text editor.

After its introduction in 1983, BSD Unix became the first widely distributed open-source version of Unix and is the basis for numerous later versions of Unix, including Darwin, the core of Mac OS X.

Bill Joy's official Sun Microsystems biography is at www.sun.com/aboutsun/media/ceo/mgt_joy.html beginning, Unix was conceived as both a multiuser and a multitasking system—that is, one that many people and many programs could use simultaneously and harmoniously.

In the late 1970s, the University of California at Berkeley used Unix extensively in its computer science department, several of whose members contributed features to the operating system. A key contribution: building in support for TCP/IP (the networking protocol suite that defines the Internet), added in the early 1980s. Virtually all current versions of Unix use the Berkeley networking code or its derivatives**.** Eventually, the version of Unix that came out of the university was dubbed the Berkeley Software Distribution (BSD)—from which the Darwin core of Mac OS X is a direct descendant.

Thus, when we refer to Mac OS X's Unix features, we're almost always talking about Darwin. And although you can run Darwin by itself, it won't look like Mac OS X without the proprietary components Apple provides.

Think of Mac OS X as having several layers: The bottom, or foundation, layer is Darwin (**Figure 1.1**). On top of Darwin are a number of proprietary software components Apple has added. Above it all is the layer users see—the graphical interface called Aqua. You can use Mac OS X for traditional Macintosh tasks without ever being aware of the layers underneath Aqua, including Darwin.

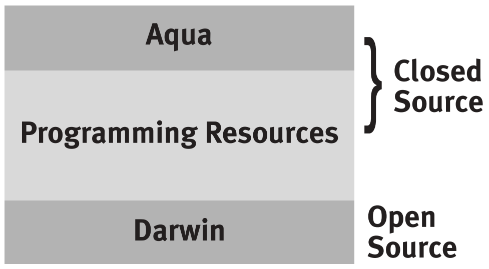

Figure 1.1 Apple built the latest version of its operating system, Mac OS X, on top of a version of Unix called Darwin. Between the Aqua interface and the underlying operating system are two layers of Apple-specific programming layers.

Unix Pioneer: Linus Torvalds

Linus Torvalds is widely known as the inventor of Linux, a completely open-source Unix-like operating system. Torvalds wrote the core of Linux, called the *kernel,* and released the first version in 1991.

Besides the Linux kernel, Torvalds's most significant contribution to Unix has been his ability to gently and productively facilitate and coordinate the efforts of literally hundreds of programmers whose work comprises the current version of the Linux kernel.

Torvalds made a key decision when he released the code for the Linux kernel under a software license called the GNU General Purpose License (GNU GPL), which requires anyone making changes to the source code to make those changes freely available to the world. Most installed versions of Linux come with hundreds of other pieces of software also licensed under the GNU GPL, and so the name GNU/Linux is usually more accurate when speaking of Linux.

The unoffical Linus FAQ is at www.tuxedo.org/~esr/faqs/linus/.

Linus's own home page is at www.cs.helsinki.fi/u/torvalds/.

Unix History Timeline

Some key dates in the development of different Unix versions:

- **1970:** Ken Thompson suggests the name *Unix* for the fledging operating system born in 1969 at AT&T Bell Labs.
- **1973:** The kernel (core) of Unix is rewritten in the C language, making it the world's first operating system that's "portable"—that is, able to run on multiple kinds of hardware.
- **1977:** First BSD (Berkeley Software Distribution) version is released. Licensees must also get a license from AT&T.
- **1983:** Version 4.2 BSD is released. By the end of 1994, more than 1,000 licenses are issued. AT&T releases its commercial version, System V.
- **1983:** AT&T releases System V release 3. IBM, Hewlett-Packard, and others base their own Unix-like systems on this version.
- **1991:** Linus Torvalds releases version 0.02 of Linux, an open-source, Unix-like operating system.
- **1992:** Bill Jolitz releases 386/BSD, a full version of Unix with no AT&T code.
- **1992:** Sun Microsystems releases Solaris, a version of Unix based on System V release 4, incorporating many BSD features.
- **1994:** BSD4.4-Lite is released by the University of California at Berkeley. It is entirely free of legal encumbrances from the old AT&T code. Version 1.0 of Linux is also released this year; Linux incorporates features from both AT&T's System V and BSD versions of Unix.
- **1999:** Apple Computer releases Darwin—a version of BSD Unix and the core of the Mac OS X.

Unix Pioneer: Dennis M. Ritchie

Dennis M. Ritchie has been a computer scientist with Bell Labs for 35 years. He is most famous for having assisted Ken Thompson in inventing Unix and for being the primary creator of the C programming language (with Brian Kernighan in 1972).

Ritchie is also a parent of another operating system, called Plan 9, that is well-known to the community of people who develop operating systems. He is currently head of the System Software Research Department at Bell Labs, where he is working on a new operating system called Inferno.

Dennis M. Ritchie's home page: www.cs.bell-labs.com/who/dmr/.

For More on Unix History

- **CrackMonkey** (http://crackmonkey.org/unix.html)—A history of Unix, including a discussion of its important flavors.
- **"The Evolution of the Unix Time-sharing System"** (http://cm.bell-labs.com/cm/cs/who/dmr/hist.html)—Coinventor Dennis M. Ritchie offers a technical and social history of Unix.
- **"Overview of the GNU Project"** (www.gnu.org/gnu/gnu-history.html)—A history of the Free Software Foundation's efforts to create an open and free version of Unix.
- **Open Source: Darwin** (www.opensource.apple.com/projects/darwin/)—Apple's official Darwin Project site, where you can download source code and find links to other related projects.
- **Darwinfo** (darwinfo.org)—General-purpose site about Darwin, including links to mailing lists.
- **The GNU-Darwin Distribution** (http://gnu-darwin.sourceforge.net)—Web site that "aims to be the most free Darwin-based Unix distribution."

Beyond Unix: Other Open Systems

Other open-source technologies are also helping to revolutionize the information infrastructure of society.

- **HTML—**The ease of creating documents using HyperText Markup Language drove the growth of the Web. Anyone viewing a Web page can see, copy, and modify the underlying HTML. The HTML standard is coordinated by the World Wide Web Consortium (www.w3c.org/MarkUp/).
- **Apache Web server** (www.apache.org)—By far the most popular Web server in the world, Apache provides a huge variety of configuration options and can be altered easily to add new ones. Mac OS X comes with Apache (see Chapter 14, "Installing and Configuring Servers").
- **Perl** (www.perl.org)—This powerful scripting and programming language (which comes with Mac OS X) is used in scripts as short as 20 lines and in large object-oriented applications with thousands of lines of code.
- **C—**The programs listed in this sidebar and those included in every version of Unix are written—with few exceptions—entirely in the C programming language. To learn more, check out *The C Programming Language,* by Brian W. Kernighan and Dennis M. Ritchie (Prentice Hall, 1998; http://cm.bell-labs.com/cm/cs/cbook/)—the fundamental book on the topic—or "The Development of the C Language" (http://cm.bell-labs.com/cm/cs/who/dmr/chist.html).
- **ssh**—The **s**ecure **sh**ell tool facilitates secure connections between computers. The open-source version is maintained by the OpenBSD project (www.openssh.org).
- g**cc**—The **GNU** **c**ompiler **c**ollection translates programming source code into machine-executable applications. It's maintained by the Free Software Foundation (www.gnu.org/software/gcc/gcc.html).
- **Sendmail** (www.sendmail.org)—Mac OS X comes with a version of this common mail server software, which you can use to set up your Mac OS X computer to be your own mail server.

How Mac OS X's Unix Differs from Mac OS 9

The immediate, obvious difference between Unix and Mac OS 9 is the user interface. Until Mac OS X, the various graphical interfaces available to Unix users all fell short of the elegance and polish to which Macintosh users are accustomed.

It is a tribute to the architecture of Mac OS X that you can use it as the next Macintosh operating system and never have to see any significant Unix underpinnings. Never, that is, unless you want to learn Unix.

From a more technical point of view, when OS X is compared with OS 9 (and earlier Macintosh operating systems), several important differences stand out. As we noted earlier, Unix uses protected memory and preemptive multitasking, and has other capabilities that let applications share memory, processors, and applications in a stable and reliable way.

As a result, it is hard for one misbehaving application to affect any other application or the operating system itself. Yes, applications can still crash in Mac OS X, but rarely do they take the whole OS down with them. Not only will you suffer fewer crashes, but they'll impact your other work less.

Unix, and thus Mac OS X, is also a multiuser operating system. It's designed from the ground up with the assumption that many people will be using the computer, often simultaneously. Just as the applications' activities are kept separate, so too are the actions of each user kept separate. Even if 50 people are using the Macintosh, it is hard for any one of them to mess up the other ones.

Another way Mac OS 9 differs from Mac OS X is the arrangement of files and folders (called *directories* in Unix) and the way information about each file is stored.

You will also see something called Home show up as a shortcut in the Finder navigation dialog box, the Save File dialog box, and so on. This concept of each user's home directory as the only place where you normally create files is a thoroughly Unix idea arising directly from Unix's nature as a multiuser system, and is a major change from Mac OS 9.

Apple has strived to mask these differences in Mac OS X, but they are still there, and you must be aware of them if you want to do serious Unix work (or even just be a Mac OS X "power user").

In your day-to-day use of Mac OS X, you'll find that many of the ways it differs from Mac OS 9 have more to do with the new Mac interface, Aqua, than with Unix. For example, the Dock is new to Mac OS X, but it isn't a "Unix change." In this book, we'll focus specifically on the Unix characteristics of Mac OS X, not the differences that come from Aqua.

What You Can Do with Mac OS X and Unix

What do you want to do? Do you want to create movies with iMovie and make them available over the Internet? With the Unix-based Mac OS X, that task is easy: You simply drag the movies into the Sites folder in your Home directory, then enable Web sharing. Would you like to run your own radio station? You can easily use your Mac OS X machine to run the Icecast server, which provides powerful streaming-audio capabilities. Do you want your schedule to be constantly available to friends and family, your résumé always accessible to potential employers? You can provide all those things with greater reliability on a Unix platform.

Beyond the foundation of increased stability, flexibility, and openness, Unix brings a number of more specific features to Mac OS X that are fundamental to the way you will use it. Key among these is the way it supports multiple users and multiple processes.

Accommodating multiple users

As we've said earlier, Unix is a multiuser environment and intentionally keeps each user's actions separate to create a more stable environment.

On a Unix system you are never alone (unless you started the machine in single-user mode, which we'll discuss in Chapter 11; if you already know what that means, keep quiet until the others have a chance to catch up). Unix assumes there are going to be many users running programs on the system.

When you log in to a Unix system, you identify yourself with a user name and password that have already been entered into the system by an administrator (if you are working on your own Mac OS X system, you will have created at least one account for yourself when you installed the operating system). This enables the operating system to keep your files and actions separate from everyone else's, and is a major factor in Unix stability and security.

✔ Tips

- You can see a list of who is logged in to your system using the command-line `w` and `who` commands. See Chapter 11, "Introduction to System Administration," for more information on using the `w` and `who` commands.
- You can see a list of all the user accounts on the system with the Netinfo Manager application (also covered in Chapter 11).

All of the files you create in the normal course of using the system are "owned" by you. Every file and every running program (known as a *process*) on a Unix system is owned by a user. All of the important system files—that is, the ones that make up the actual operating system—are owned by a special super-user called *root.* The root account is all-powerful, and you must exercise great care when using it (see Chapter 11 for more on root).

Not every user account on the system is intended for use by a human. Unix systems, including Mac OS X, come with a number of special user accounts with names like "nobody" and "daemon." The system uses these accounts to own processes that should not have the power of the root account.

Each regular user account on a Unix system has its own area on the file system called its *home directory.* This is where all of the files a given user creates and owns are stored. Unix keeps track of who owns each file and allows (or disallows) various operations based on the ownership of files.

Preemptive multitasking

On Unix systems, you might not only have multiple programs running, but you might also have multiple copies of the same program running. Even with just a single person logged in, running a few applications, several dozen processes will be running at any given moment, each with its own separate memory allocation. In fact, the operating system keeps a number of different processes running even if you are not doing anything.

When the machine starts up, the initial process (called *init*), which is owned by the super-user root, begins. The init process then starts many other processes, which are also owned by root.

✔ Tip

- You can see a list of all the processes on your system using the `ps` and `top` commands. See Chapter 11 for more information on monitoring system usage.

Applications vs. Programs

All applications are programs—the terms are synonymous. In this book we use the term *application* to refer to complex programs used for a variety of related tasks—for example, Adobe Photoshop is an application for graphics manipulation. In Unix you often see the term *command,* which can refer either to an application that handles some specific task (such as copying files) or to a built-in feature of a larger program or application. For example, the command for copying files is the `cp` command, which is in fact a small program. The command to move from one folder into another folder (*directory* in Unix terms) is the `cd` command, which is actually part of a larger program called the shell. See Chapter 5, "Using Files and Directories," for more on the `cp` and `cd` commands.

Parents and children

Every process in Unix is the child of some other process, except for that first process, init, which is the mother of all processes. This concept of processes having parents and children comes up frequently in Unix.

When you log in to a Unix system, you start a process that you alone own. The exact program depends on which Unix system you are using, and how you log in to it. This process will be the parent (or grandparent, or great-grandparent) of every process you start on the system.

When you log in using the Mac OS X graphical user interface, you start a process called WindowServer, which you own.

Every program you run will have the WindowServer process as an ancestor. In other words, if you start up an application such as BBEdit, a popular text editor, then BBEdit's parent process will be WindowServer. If you start up the command-line interface (the Terminal application), you might then start more programs using the Terminal application. Those programs will have Terminal as their parent, and WindowServer as their grandparent, and so on.

So, using your Mac as a single-user system, you might have several dozen processes running.

Files and the filesystem

Unix brings a number of changes to the Mac OS with regard to files and the filesystem (see the sidebar "What Is a Filesystem?" for its definition).

From the user's point of view, the most prominent changes (as compared with Mac OS 9) involve the handling of file security, storage of files, and the use of a different syntax for describing a file's location.

Files and security

On an old Mac OS system, you could alter or delete any file. You could put files from the System Folder in the Trash and cause all kinds of trouble, even accidentallly. On a Unix system, every file is owned by some user. The operating system restricts the ability to create, change, or delete files based on ownership, so that one user cannot alter or delete files created by another user, and you are unlikely to cause any serious damage to the operating system (the exception: the root user can do anything).

Folders are called directories

What Mac users call a *folder* Unix users call a *directory.* A directory that is inside another is called a *subdirectory.* It is important to know which directory you are "in," because when you're working from a command line there is no visual cue, such as an active window. Know the concept of the "current directory" in Unix—that's where you're currently working.

File paths use / instead of :

In Mac OS 9 and earlier, file path designations use colons. In Mac OS X, Unix uses the / (slash) instead of the : (colon) to separate the parts of a file path. In Mac OS 9, then, the path of the FileMaker Pro application would look like that in **Figure 1.2**.

In Mac OS X, the same path would appear as shown in **Figure 1.3**.

Notice that the name of the hard drive doesn't show up anymore. In Unix, drives don't have names. The Mac OS X Aqua interface does have names for drives, and they do show up in the Finder, thanks to some tricks of the Mac OS X Finder, but at the underlying Unix/Darwin level, even in Mac OS X, drives don't have names.

Unix treats the entire filesystem as if it were one big disk. There are ways to see which disks contain which files, but usually when dealing with files in Unix, you only pay attention to the *full pathname* of the file. See Chapter 5, "Using Files and Directories," for more about using pathnames.

Even with the differences cited above, the Unix filesystem is organized similarly to what you are used to on a Mac. Use / instead of : in your pathnames, and think of your system as a Macintosh that has only one disk. Think of that disk as being named /, and then you're close to the way Unix thinks of files, directories, and subdirectories.

```
MyBig Disk:Applications (Mac OS 9):FileMaker Pro 5 Folder:FileMaker Pro
```

Figure 1.2. This is how the path of the FileMaker Pro application would look on the Mac OS 9 file system.

```
/Applications (Mac OS 9)/FileMaker Pro 5 Folder/FileMaker Pro
```

Figure 1.3 This is how the path of the FileMaker Pro application would look on the Mac OS X filesystem.

What Is a Filesystem?

In Unix the term *filesystem* (Unix's terminology for file system) is used in two ways. The first way is more informal and refers to the complete hierarchy of directories. The second way refers to a single storage area that has been formatted for use by the operating system. The "single storage area" is often, but not always, a single disk partition. Filesystems contain directories and files but never other filesystems.

Example of the first form: "/ is the root directory of the filesystem."

Example of the second form: "It is common to have two or more filesystems on the same physical disk."

Get used to filename extensions

Another difference between Mac OS 9 and Mac OS X is in OS X's use of filename extensions—you know, those things at the end of all the filenames on the Web and on PCs, such as .html, .txt, and .jpg (this is no surprise given which operating system the original Web servers used and the one most Web servers still use today).

From the very start, the Mac OS has cleverly kept track of a file's characteristics: what type of file it is, what application opens it, if the file is being used by another application, if the file is locked. Unix doesn't store as much information about each file along with each file. In particular, Unix has no fundamental concept of a file's "type" or "creator" (Mac OS X does, but only for files that were created with Mac file information).

Unix's filename extensions indicate a file's type. This is not as powerful as the Macintosh approach, but it is the standard in the Unix world. Mac OS X tries to have it both ways, and in the Mac spirit uses the old Mac approach in some cases and the standard Unix approach in other cases. But in order to play well with others, Mac OS X incorporates filename extensions. You can decide whether to display them in the graphical interface, but when you use the command line they will always be there.

Files created by Macintosh applications will have the Macintosh creator and type attributes, but files created by non-Mac applications, including all non-Mac Unix applications, will have only the filename extension (if any) to indicate what kind of file they are.

How You Will Be Working with Unix

You are probably already using your Macintosh for a variety of tasks, working in applications that take advantage of the lovely Aqua interface. As you dig below the surface and start using the Darwin layer of Mac OS X, you will be performing operations that are either unique to Unix or better suited to the Unix environment.

Working from the command line

The command line is the primary user interface in Unix. Most Unix software packages are designed to be installed and configured from the command line.

It is from the command line that you will be installing software and manipulating files (copying, moving, renaming, and so on). You might even start editing files using the command-line tools.

One of the most powerful aspects of the command line is in how it allows you to connect a series of commands together to accomplish some task. **Figure 1.4** shows an example of connecting three commands together in order to find all the files in a folder that contain the word *success* and email the resulting list of filenames to yourself.

The command line in Figure 1.4 is composed of three major parts separated by the vertical bar (`|`) character. (Note: this command line requires that you have activated the email server as described in Chapter 14, "Installing and Configuring Servers.")

The first part uses the `find` command to produce a list of the names of all the files (not folders) in the current folder (by using the `-type f` option) and all those inside it. The output of that command is passed (*piped*) via the `|` character to the next command, `xargs` (*arguments*). This applies the `grep` (*search*) command to each filename in turn, searching the file for the string "success" and producing a list of the filenames where the string was found. That second list is piped to the third part, the `Mail` command, which sends the list to the specified email address. The final ampersand (`&`) tells Unix to do all this "in the background," which means that we do not have to wait for the processes to finish before issuing a new command—we can go on with our work at the command line. Chapter 2, "Using the Command Line," takes you further into the details of using the command line.

Under Mac OS X, the most common way to get to the command line is through the Terminal application (found in the Utilities folder under Applications).

```
find . -type f -print0 | xargs -0 grep -l success | Mail address@hostname.com &
```

Figure 1.4 This command line shows how to connect commands together—in this case, finding all the files in a folder that contain the word *success* and then emailing the resulting list of file name to yourself.

Editing files from the command line

In order to really harness the power of Unix, you will want to learn how to edit files using a command-line text editor. Unix is file-centric and uses text files to control almost every aspect of software configuration. Although it's difficult for most Mac users to learn at first, editing files from the command line lets you change files without leaving the command-line environment in which most of your Unix work will occur. Furthermore, the ability to edit files from the command line will make it easy for you to work on other Unix systems besides Mac OS X, something you are almost certain to do once you get further into Unix.

Unix Commands Have Strange Names

Unix commands often have very terse obscure and/or arbitrary-sounding names. Examples: `awk`, `grep`, and `chmod`.

This contributes to Unix's (justly earned) reputation as a difficult operating system to use, requiring users to memorize a great deal in order to become proficient.

Because you are probably itching to know how those three commands got their names, here's the story: `awk`, a text-processing system, got its name from the initials of the three people who created it. `grep`, a command for searching inside text, got its name from the commands used in an earlier program to "globally find a regular expression and print." `chmod` is a command to change the permissions associated with a file and means "CHange MODe."

Programming and scripting

Developed *by* programmers *for* programming, Unix is—not surprisingly—an excellent programming environment, and many of its strengths (and some of its weaknesses) stem from that heritage.

Although you don't need Mac OS X or Unix to create software, if you're using Unix, you'll probably at least poke around with programming—perhaps first modifying existing programs and then moving on to create new ones. In addition to its terrific stability, Unix provides an environment in which it's easy to connect varying tools in an equally various number of ways. And when you need them, you can create new commands, extending your tool kit as you work.

You can also write simple scripts to automate tasks—for example, to perform backups, automate the transfer of files to other systems, calculate the rate of return on an investment, or search text for certain phrases and highlight them. You could write scripts to create a small database-backed Web site, or to convert batches of images for use on the Web, or to analyze voter-registration or campaign-contribution records. Some users never stop creating new applications: We call them *programmers.*

Mac OS X comes with tools to create and run programs in AppleScript, Perl, Bourne shell, and a couple of other Unix scripting languages**.** The Mac OS X developer tools include software that allows you to create programs in C, C++, Objective-C, and Java as well. (Throughout this book we assume that you have in fact installed the Developer Tools.) With the exception of AppleScript, none of these programming languages were available to Mac users in the past unless they installed third-party software (such as MacPerl or the CodeWarrior compiler). The Mac OS X

continues on next page

Developer Tools also include the Project Builder and Interface Builder applications, which are graphical interfaces for developing software projects written in C, Objective-C, C++, and Java.

Shell scripts

The vast majority of Unix scripting is done using *shell scripts*. These are written using the language of a Unix *shell*. A Unix shell is the program that provides the command-line interface you will be using. A shell accepts typed commands and provides output in text form; it is a "shell" around the operating system. The Bourne shell is one of the oldest command-line interpreters for Unix (see Chapter 2, "Using the Command Line"). Virtually all of the scripts that control what happens when Unix machines start up are written in the Bourne shell scripting language, including most of the Mac OS X startup files.

If you are excited or impatient, you probably want to take a look at one of the Mac OS X system startup scripts right now! Here's how to do it:

To view a system startup script:

1. Open an OS X (not Classic) text editor—for example, the Textedit application, which you can access through Textedit in the Applications folder.
2. Open the file `/System/Library/StartupItems/Network/Network.`

   The file will be opened read-only, so you need not worry about damaging it.

You are looking at the script that configures your network connection on startup (**Figure 1.5**).

```
#!/bin/sh

##
# Apache HTTP Server
##

. /etc/rc.common

StartService ()
{
    if [ "${WEBSERVER:=-NO-}" = "-YES-" ]; then
        ConsoleMessage "Starting Apache web server"
        apachectl start
    fi
}

StopService ()
{
    ConsoleMessage "Stopping Apache web server"
    apachectl stop
}

RestartService ()
{
    if [ "${WEBSERVER:=-NO-}" = "-YES-" ]; then
        ConsoleMessage "Restarting Apache web server"
        apachectl restart
    else
        StopService
    fi
}

RunService "$1"
```

Figure 1.5 The script `/System/Library/StartupItems/Network/` Network configures your network connection on startup.

Figure 1.6 on the next page is an example of a script you might use in Mac OS X to make a group of files open in Photoshop when they're double-clicked from the Finder. Using this script and some additional Unix commands, you could instruct your machine to find every file ending in .jpg within a directory (folder) and have those files launch Photoshop when double-clicked from the Finder. And by altering the script, you could do the same thing for just those files that already have the Mac type code for JPEGs, GIFs, and others. (The

```
#!/bin/sh
# This a comment. Comments help make the code easier to read.
# This script takes one or more file names as arguments and
# sets the Creator Code for each one to Photoshop.

GETINFO="/Developer/Tools/GetFileInfo"
SETFILE="/Developer/Tools/SetFile"

#  8BIM is the Creator Code for Photoshop
NEW_CREATOR="8BIM"
changed_files=0
total_files=0
for file in "$@" ;   # All the command-line arguments are in $@
do
    total_files=`expr $total_files + 1`;  # keep track of total
if [ -w "$file" ];  # If the file is writeable...
    then
        creator=`$GETINFO -c "$file"`; # Get the Creator code of this file
        if [ !  "$creator" = \"$NEW_CREATOR\" ]
        then
            # Set the file to have the new creator code
            $SETFILE -c "$NEW_CREATOR" "$file"
            changed_files=`expr $changed_files + 1`
        fi
    else
        echo "skipping '$file' - not writeable"
    fi
done

echo "Checked $total_files files"
echo "Set $changed_files files to have creator $NEW_CREATOR"
skipped=`expr $total_files - $changed_files`
echo "Skipped $skipped files"
```

Figure 1.6 You might use this script in Mac OS X to make a group of files open in Photoshop when they're double-clicked from the Finder.

type code is a four-character code that identifies the type of each file. It's a pre–Mac OS X feature that many Mac applications still use.)

This example may look scary now, but don't worry—once you learn some Unix, it will make more sense. For now, just let it wash over you, and understand that when you've read this book (and thus know a bit of Unix), you'll be able to create this sort of script fairly easily. (See Chapter 9, "Creating and Using Scripts," for more on creating shell scripts.)

Perl

Perl is one of the most popular programming languages in the world. Although you can use it to build large, complex programs, it is easy enough to learn that most people begin using it to write small utility programs or CGI programs for Web sites.

Because Perl excels at text processing and can easily interact with SQL databases, it's ideal for building Web pages as well as other data-manipulation projects.

Figure 1.7 is a code listing of a Perl script that outputs plain text files in reverse—the last line comes out first. This would be difficult, if not impossible, using traditional Macintosh applications.

Java

Although still fairly new, Java has already spread far and wide—partly because it's powerful and partly because its creator, Sun Microsystems, has promoted it very hard.

Programs written in Java can run on only one kind of machine, but that machine is a *virtual machine*—a piece of software. Because a virtual machine is software, it can be written for different hardware platforms. Java virtual machines exist for every major operating system, and an increasing number of small hardware devices (such as cell phones) are able to run Java code. Programs written in Java can often run without changes on many different platforms. The Mac OS X Developer Tools come with a Java compiler and a Java virtual machine. The Java programming language has a large set of tools for creating graphical user interfaces and for communicating across networks.

```
#!/usr/bin/perl
# This a comment. Always use comments.
#
# This script takes one or more file names as arguments and
# outputs the files one line at a time, in reverse order.
# I.e. The last line of the last file comes out first.

while ( $file =  pop(@ARGV) ) {
    open FILE, "$file"; # Open the file for reading
    @lines = <FILE>;    # Read the entire file into @lines
    close FILE;
    while ( $line = pop(@lines) ) {
        print $line;
    }
}
```

Figure 1.7 This Perl script code listing outputs plain text files in reverse, with the last line first.

C

The C programming language is to programming what Greek is to literature—the language of heroes. C is the language in which Unix as we know it was written, and most of the utility programs used with Unix were written in C. In the Unix world, the people who invented Unix could be thought of as heroes, and they wrote their great works in C.

The core of every Unix operating system (the *kernel)* is written almost entirely in C, as is virtually every common Unix utility program, such as `ls`, `pwd`, and `grep`. Many important Unix applications, such as Sendmail and the Apache Web server, are also written in C. In addition, C++, Objective-C, and a number of other important languages stem from or are related to C.

Because so much Unix software is written in C, you're likely to at least modify existing C code if you spend much time working on the Unix platform.

Interacting with other Unix machines

Much of what people do with their Unix systems involves connecting to other Unix systems—for example, logging on to a machine that hosts a Web site to edit files and install software, or arranging to automatically transfer files between two Unix machines.

To ease this process, Mac OS X comes with a widely used program called `ssh` (*S*ecure *sh*ell) that facilitates secure (encrypted) connections to other machines over the Internet. With ssh you can connect from the command-line interface to other Unix machines and the information exchanged is protected from being read if intercepted. Other programs also use ssh to work over encrypted connections.

Running servers

One of the biggest differences between Mac OS 9 and Mac OS X is that the latter allows you to *reliably* run servers (such as a Web server or an email server) on your computer. You could run these types of servers on Mac OS 9, but because OS 9 was much more likely to crash, you probably wouldn't consider it for serious use. Also, most of the software available for these kinds of servers on Mac OS 9 was closed-source proprietary software, so if the vendor changed its business plan or went out of business, you were left with unsupported software. With Mac OS X, you can use the widely installed, open-source applications that most servers on the Internet use.

You might run a server to provide a service to the rest of the world. Or you might run one because you're developing a system that uses it—for example, a shared calendar/event-planning system—and you want to test it on your local machine and/or network before deploying it. There are all kinds of servers; the following are just a few of those available to you as a Mac OS X user.

Apache Web server

Apache is the most popular Web server in the world—that is, more Web sites use Apache servers than any other. Apache is highly configurable, so it can be adapted to many different situations and specific requirements. Apache comes installed in Mac OS X.

Icecast audio server

Icecast (www.icecast.org) is an open-source audio-streaming system for the Internet. Fairly easy to install and configure, Icecast allows you basically to turn your Mac into an Internet radio station.

Using other Unix applications

In addition to the specific applications mentioned above, there are thousands of Unix applications available. Most are free, some are commercial packages, some are open source. With Mac OS X you can use many of these existing applications to monitor network status, analyze data such as Web-server log files, run mailing lists, create Web publishing systems, and more.

Because there are so many, we can't list even a tenth of them. Below are a few that give some sense of the variety available, plus links to places where you can find more.

Samba Windows file-sharing software

Samba lets you share files from your Macintosh with Windows users over a network. The name Samba comes from SMB (Server Message Block), which is the Windows file-sharing protocol.

SQL database engines

If you are a Mac database user, you have heard of FileMaker Pro, which is a great database with a great user interface. But FileMaker Pro doesn't understand SQL (Structured Query Language), which is what all the big serious databases use. Most database-backed Web sites use SQL databases.

A number of SQL database engines are available for Mac OS X, including MySQL, PostgreSQL, and ProSQL.

Image manipulation with GIMP

GIMP (GNU Image Manipulation Program) (www.gimp.org) is a bargain-basement Unix version of Photoshop. Even though GIMP is not as powerful as the main commercial alternative, it is free, open-source software, and runs on many Unix platforms. Using GIMP requires that you install X Windows (see below).

X Windows

X Windows (www.osxgnu.org/software/Xwin/) is the underlying mechanism for providing a graphical user interface on most Unix systems. Mac OS X uses a different method, Apple's proprietary Aqua interface, but many Unix programs were built for X Windows, so if you install it, you can use these other programs (such as GIMP).

You can use X Windows to provide a graphical display for Unix programs that are running on other machines over the Internet. That is, if you are running X Windows on your Mac, and you have an account on another Unix machine somewhere on the Internet, you may be able to run software on the remote machine and see the graphical display on your Mac.

Email list management with Majordomo

Majordomo (www.greatcircle.com/majordomo/) is a free, open-source application for managing multiple email. To use it, your Macintosh must be set up as an email server. With Majordomo, you can run dozens of mailing lists, with different configurations for each one. For example, one list may require that new subscribers be added by the list owner, while another may allow anyone to self-subscribe via email. Majordomo, which was written in Perl, supports list archives and digests as well as many other features.

Where to find more

Thousands of Unix programs are available, with more being created every day. A couple of places to look:

The FreeBSD Ports Collection

This collection (www.freebsd.org/ports/) offered more than 6,700 open-source applications as of spring 2002. These are all Unix programs that work on a number of different Unix versions. Because the Darwin layer of Mac OS X is based largely on FreeBSD, most of these programs should work on Mac OS X.

The easiest way to install many of the FreeBSD programs (and other Unix programs) on Mac OS X is to use the Fink program, which is covered in Chapter 13, "Installing Software from Source Code."

Mac OS X Apps

This Web site (www.macosxapps.com) provides a large and growing collection of Mac OS X applications, most of which have graphical interfaces and can be installed in a manner familiar to Mac users. Many of these programs are not "pure" Unix programs in that they make use of proprietary Mac OS X features such as the Aqua interface. Still, many take advantage of the Unix core of Mac OS X, and so this site is a good place to explore.