1

Unix Fundamentals

The Unix operating system was created more than 30 years ago by a group of researchers at AT&T's Bell Laboratories. During the three decades of constant development that have followed, Unix has found a home in many places, from the ubiquitous mainframe to home computers to the smallest of embedded devices. This chapter provides a brief overview of the history of Unix, discusses some of the differences among the many Unix systems in use today, and covers the fundamental concepts of the basic Unix operating system.

Brief History

In terms of computers, Unix has a long history. Unix was developed at AT&T's Bell Laboratories after Bell Labs withdrew from a long-term collaboration with General Electric (G.E.) and MIT to create an operating system called MULTICS (Multiplexed Operating and Computing System) for G.E.'s mainframe. In 1969, Bell Labs researchers created the first version of Unix (then called UNICS, or Uniplexed Operating and Computing System), which has evolved into the common Unix systems of today.

Unix was gradually ported to different machine architectures from the original PDP-7 minicomputer and was used by universities. The source code was made available at a small fee to encourage its further adoption. As Unix gained acceptance by universities, students who used it began graduating and moving into positions where they were responsible for purchasing systems and software. When those people began purchasing systems for their companies, they considered Unix because they were familiar with it, spreading adoption further. Since the first days of Unix, the operating system has grown significantly, so that it now forms the backbone of many major corporations' computer systems.

> *Unix no longer is an acronym for anything, but it is derived from the UNICS acronym. Unix developers and users use a lot of acronyms to identify things in the system and for commands.*

Unix Versions

In the early days Unix was made available as source code rather than in the typical binary form. This made it easier for others to modify the code to meet their needs, and it resulted in forks in the code, meaning that there are now many disparate versions (also known as flavors).

> *Source code represents the internal workings of a program, specifying line by line how a program or application operates. Access to source code makes it easier to understand what is occurring in the program and allows for easier modification of the program. Most commercial programs are distributed in binary form, meaning they are ready to be run, but the internal lines of code are not readable by people.*

There are primarily two base versions of Unix available: AT&T System V and Berkley Software Distribution (BSD). The vast majority of all Unix flavors are built on one of these two versions. The primary differences between the two are the utilities available and the implementations of the file structure. Most of the Unix flavors incorporate features from each base version; some include the System V version utilities in `/usr/bin` and the BSD version in `/usr/ucb/bin`, for example, so that you have the choice of using a utility with which you are comfortable. This arrangement is indicative of the Unix way of providing the flexibility to do things in different ways.

The various versions of Unix systems provide the user the power of choice: you can select the flavor that best matches your needs or system requirements. This ability to choose is considered by many as a strength, although some see it as a weakness in that these slightly differing versions and flavors create some incompatibilities (in the implementation, commands, communications, or methods, for example). There is no "true" version of Unix or one that is more official than others; there are just different implementations. Linux, for example, is a variant of Unix that was built from the ground up as a free Unix-like alternative to the expensive commercial Unix versions available when Linux was first created in 1991. Here are some of the more popular flavors of Unix available:

Sun Microsystem's Solaris Unix
IBM AIX
Hewlett Packard HP-UX
Red Hat Enterprise Linux
Fedora Core
SUSE Linux
Debian GNU/Linux
Mac OS X
KNOPPIX
Yellow Dog Linux (for Apple systems)
Santa Cruz Operations SCO OpenServer
SGI IRIX
FreeBSD
OpenBSD
NetBSD
OS/390 Unix
Plan 9

Each of these flavors implements its version of Unix in a slightly different way, but even though the implementation of a command may vary on some systems, the core command and its functionality follow the principles of one of the two major variations. Most versions of Unix utilize SVR4 (System V) and add the BSD components as an option to allow for maximum interoperability. This is especially true with commands; for example, there are two versions of the `ps` command (for showing processes) available on most systems. One version of `ps` might reside in `/usr/bin/ps` (the System V version) while the other might exist in `/usr/ucb/bin` (BSD version); the commands operate similarly, but provide output or accept optional components in a different manner.

Many vendors have attempted to standardize the Unix operating system. The most successful attempt, a product of the noncommercial Institute for Electrical and Electronics Engineers, is standard 1003 (IEEE 1003), also known as the POSIX (Portable Operating Systems Interface) standard. That standard is also registered with the International Organization for Standardization under ISO/IEC 9945-1, which you can find at `http://iso.org/iso/en/CombinedQueryResult.CombinedQueryResult?queryString=9945`. The POSIX standard merged with the Single Unix Specification (SUS) standard to become one integrated standard for all Unix flavors. It retained the name POSIX standard. Not all Unix versions follow the POSIX standard to the letter, but most do adhere to the major principles outlined in the standard.

Early Unix systems were mainly commercial commodities like most software for sale; to run the operating system, you generally had to pay for that right. In 1984 an engineer named Richard Stallman began work on the GNU Project, which was an effort to create an operating system that was like Unix and that could be distributed and used freely by anyone. He currently runs the Free Software Foundation (`http://gnu.org/fsf/fsf.html`), and many of the programs he and his supporters have created are used in both commercial and open-source versions of Unix.

> *GNU stands for GNU's Not Unix, which is a recursive acronym. The GNU Project wanted to create a Unix-like operating system, not a Unix derivative (which would imply that it was a source-code copy of Unix).*

In 1991 Linus Torvalds, a Finnish graduate student, began work on a Unix-like system called Linux. Linux is actually the kernel (kernels are discussed later in this chapter), while the parts with which most people are familiar — the tools, shell, and file system — are the creations of others (usually the GNU organization). As the Linux project gained momentum, it grew into a major contender in the Unix market. Many people are first introduced to Unix through Linux, which makes available to desktop machines the functionality of a Unix machine that used to costs thousands of dollars. The strength of Linux lies in its progressive licensing, which allows for the software to be freely distributable with no royalty requirements. The only requirement for the end user is that any changes made to the software be made available to others in the community, thus permitting the software to mature at an incredibly fast rate. The license under which Linux is distributed is called the GNU Public License (GPL), available at `http://gnu.org/licenses/licenses.html`.

Another free variant of Unix that has gained popularity is the BSD family of software, which uses the very lenient BSD License (`http://opensource.org/licenses/bsd-license.php`). This license allows for free modification without the requirement of providing the software source code to others. After a landmark 1994 lawsuit settlement, BSD Unix became freely distributable and has evolved into the NetBSD, FreeBSD, and OpenBSD projects, and it also forms the underlying technology for Darwin (upon which Mac OS X is based).

These freely available Unix derivatives have given new life to the Unix operating system, which had been experiencing a decline as the Microsoft Windows juggernaut advanced. Additionally, Apple has become the highest-volume supplier of Unix systems. Now Unix is moving forward in the corporate environment as well as in the end-user desktop market.

Operating System Components

An operating system is the software interface between the user and the hardware of a system. Whether your operating system is Unix, DOS, Windows, or OS/2, everything you do as a user or programmer interacts with the hardware in some way. In the very early days of computers, text output or a series of

lights indicated the results of a system request. Unix started as a command-line interface (CLI) system — there was no graphical user interface (GUI) to make the system easier to use or more aesthetically pleasing. Now Unix has some of the most customizable user interfaces available, in the forms of the Mac OS X Aqua and Linux's KDE and GNOME interfaces among others, making the Unix system truly ready for the average user's desktop.

Let's take a brief look at the components that make up the Unix operating system: the kernel, the shell, the file system, and the utilities (applications).

Unix Kernel

The kernel is the lowest layer of the Unix system. It provides the core capabilities of the system and allows processes (programs) to access the hardware in an orderly manner. Basically, the kernel controls processes, input/output devices, file system operations, and any other critical functions required by the operating system. It also manages memory. These are all called autonomous functions, in that they are run without instructions by a user process. It is the kernel that allows the system to run in multiuser (more than one user accessing the system at the same time), multitasking (more than one program running at a time) mode.

A kernel is built for the specific hardware on which it is operating, so a kernel built for a Sun Sparc machine can't be run on an Intel processor machine without modifications. Because the kernel deals with very low-level tasks, such as accessing the hard drive or managing multitasking, and is not user friendly, it is generally not accessed by the user.

One of the most important functions of the kernel is to facilitate the creation and management of processes. Processes are executed programs (called jobs or tasks in some operating systems) that have owners — human or systems — who initiate their calling or execution. The management of these can be very complicated because one process often calls another (referred to as *forking* in Unix). Frequently processes also need to communicate with one another, sending and receiving information that allows other actions to be performed. The kernel manages all of this outside of the user's awareness.

The kernel also manages memory, a key element of any system. It must provide all processes with adequate amounts of memory, and some processes require a lot of it. Sometimes a process requires more memory than is available (too many other processes running, for example). This is where virtual memory comes in. When there isn't enough physical memory, the system tries to accommodate the process by moving portions of it to the hard disk. When the portion of the process that was moved to hard disk is needed again, it is returned to physical memory. This procedure, called *paging*, allows the system to provide multitasking capabilities, even with limited physical memory.

Another aspect of virtual memory is called *swap*, whereby the kernel identifies the least-busy process or a process that does not require immediate execution. The kernel then moves the entire process out of RAM to the hard drive until it is needed again, at which point it can be run from the hard drive or from physical RAM. The difference between the two is that paging moves only part of the process to the hard drive, while swapping moves the entire process to hard drive space. The segment of the hard drive used for virtual memory is called the *swap space* in Unix, a term you will want to remember as you move through this book. Running out of swap space can cause significant problems, up to and including system failure, so always be sure you have sufficient swap space. Whenever swapping occurs, you pay a heavy price in significantly decreased performance, because disks are appreciably slower than physical RAM. You can avoid swapping by ensuring that you have an adequate amount of physical RAM for the system.

Shells

The shell is a command line interpreter that enables the user to interact with the operating system. A shell provides the next layer of functionality for the system; it is what you use directly to administer and run the system. The shell you use will greatly affect the way you work. The original Unix shells have been heavily modified into many different types of shells over the years, all with some unique feature that the creator(s) felt was lacking in other shells. There are three major shells available on most systems: the Bourne shell (also called sh), the C shell (csh), and the Korn shell (ksh). The shell is used almost exclusively via the command line, a text-based mechanism by which the user interacts with the system.

The Bourne shell (also simply called Shell) was the first shell for Unix. It is still the most widely available shell on Unix systems, providing a language with which to script programs and basic user functionality to call other programs. Shell is good for everyday use and is especially good for shell scripting because its scripts are very portable (they work in other Unix versions' Bourne shells). The only problem with the Bourne shell is that it has fewer features for user interaction than some of the more modern shells.

The C shell is another popular shell commonly available on Unix systems. This shell, from the University of California at Berkeley, was created to address some of the shortcomings of the Bourne shell and to resemble the C language (which is what Unix is built on). Job control features and the capability to alias commands (discussed in Chapter 5) make this shell much easier for user interaction. The C shell had some early quirks when dealing with scripting and is often regarded as less robust than the Bourne shell for creating shell scripts. The quirks were eventually fixed, but the C shell still has slight variations, resulting from different implementations based on which entity (commercial provider or other resource) is providing the shell.

The Korn shell was created by David Korn to address the Bourne shell's user-interaction issues and to deal with the shortcomings of the C shell's scripting quirks. The Korn shell adds some functionality that neither the Bourne or C shell has while incorporating the strong points of each shell. The only drawback to the Korn shell is that it requires a license, so its adoption is not as widespread as that of the other two.

These are by no means the only shells available. Here's a list of some of the many shells available for the different Unix systems:

- sh (also known as the Bourne shell)[
- PDKSH (Public Domain Korn shell)
- bash (Bourne Again Shell — a revamped version of Bourne shell)
- Z shell
- TCSH (TENEX C shell)

As with everything Unix, there are many different implementations, and you are free to choose the shell that best suits your needs based on the features provided. Chapter 5 examines several shells in detail.

The Other Components

The other Unix components are the file system and the utilities. The file system enables the user to view, organize, secure, and interact with, in a consistent manner, files and directories located on storage devices. The file system is discussed in depth in Chapter 4.

Utilities are the applications that enable you to work on the system (not to be confused with the shell). These utilities include the Web browser for navigating the Internet, word processing utilities, e-mail programs, and other commands that will be discussed throughout this book.

Try It Out Run Unix from a CD-ROM

The best way to learn Unix is to follow along with the book and try some of the exercises while you are reading. If you don't have a current install of a Unix operating system, and you do have an Intel/AMD-based system (a PC that is Windows compatible), you can use KNOPPIX, a bootable Linux distribution. KNOPPIX enables you to try Unix right from a CD, without installing or modifying any other operating system on your computer. It provides a full-featured Linux environment and is a great way to see what Linux and Unix is about.

1. Use the copy of Knoppix included on this book's CD or download the KNOPPIX ISO image from one of the mirrors listed at `http://knopper.net/knoppix-mirrors/index-en.html`. There are usually two versions of the software, one in German and one in English; choose the image with the extension `-EN.iso`.
2. If you downloaded a copy of Knoppix, use your favorite CD-burning software to burn a copy of the ISO onto a CD-R.
3. Insert the CD-ROM included with this book or the CD-R you created into your CD-ROM drive and boot (load) from it. By default, most systems let you boot from a CD-ROM simply by putting the disk in the drive. (If the CD-ROM doesn't start automatically, you may need to contact your computer manufacturer's manual for instructions.) You'll see the opening KNOPPIX screen, which should be similar to the one in Figure 1-1.

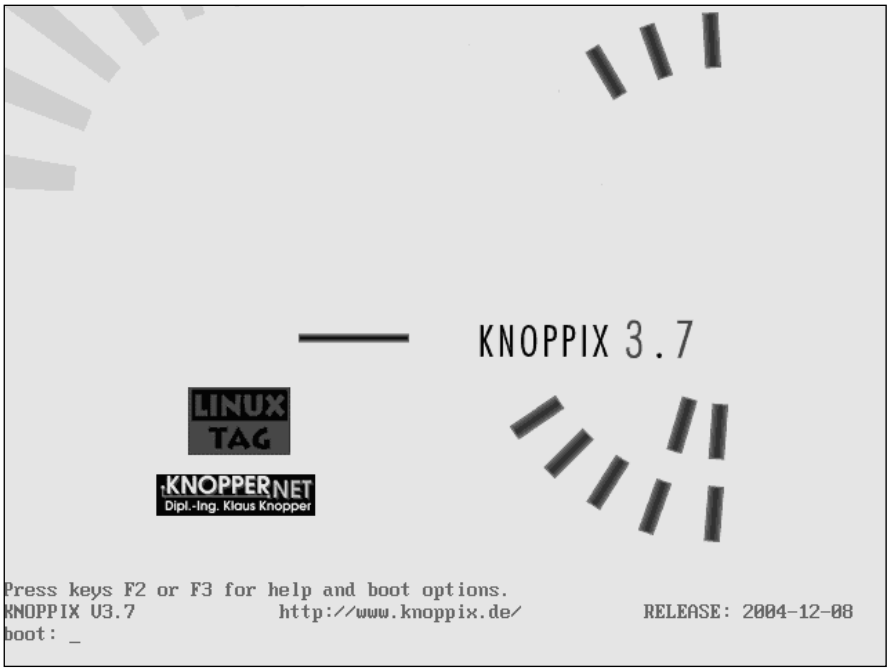

Figure 1-1

4. Press **Enter** (or **Return**) to continue the boot process. You'll see a screen similar to the one shown in Figure 1-2.

Figure 1-2

5. The boot sequence continues through a few more screens.

Because KNOPPIX is bootable and can be transported from system to system, you do not enter a password as you would with most Unix distributions.

Figure 1-3 shows the desktop loading.

6. When you are done, exit the system by rebooting (restarting) or shutting down your computer. You can do this by pressing **Ctrl+Alt+Del**. A dialog box provides you with options to Turn Off Computer or Restart Computer. If you select Restart Computer, take out the CD-ROM during the reboot to return to your regular operating system.

Figure 1-3

How It Works

The KNOPPIX distribution has been optimized to run within RAM from the CD-ROM. It does not need to modify the hard drive or install itself anywhere. It can be run without fear of damaging the current contents of your hard drive.

Summary

This chapter briefly discussed the history of Unix and introduced some of the versions of Unix. The Unix core components — the kernel, shells, file system, and utilities — were introduced.

In the past, Unix was considered a system geared to the most computer-savvy users and those who wanted a system for core functionality, with no regard to aesthetics or user friendliness. Unix has evolved to fit the needs of many different types of users, from the no-nonsense corporate environment to the novice computer user's desktop. There are rich desktop environments available for many flavors of Unix, for example, and every currently selling Macintosh computer is running a version of Unix right out of the box.

In Chapter 2, you begin using a Unix system from initial login to logout.