Chapter 1. Introduction to Linux

1.0 Introduction

Linux is a fast and stable open source operating system for personal computers (PCs) and work stations that features professional-level Internet services, extensive development tools, fully functional graphical user interfaces (GUIs), and a massive number of applications ranging from office suites to multimedia applications. Linux was developed in the early 1990s by Linus Torvalds, along with other programmers around the world. As an operating system, Linux performs many of the same functions as Unix, Macintosh, Windows and Windows NT. However, Linux is distinguished by its power and flexibility, along with being freely available. Most PC operating systems, such as Windows, began their development within the confines of small, restricted PCs, which have only recently become more versatile machines. Such operating systems are constantly being upgraded to keep up with the ever-changing capabilities of PC hardware. Linux, on the other hand, was developed in a different context. Linux is a PC version of the Unix operating system that has been used for decades on mainframes and minicomputers and is currently the system of choice for network servers and workstations. Linux brings the speed, efficiency, scalability, and flexibility of Unix to your PC, taking advantage of all the capabilities that PCs can now provide.

Technically, Linux consists of the operating system program referred to as the kernel, which is the part originally developed by Linus Torvalds. But it has always been distributed with a massive number of software applications, ranging from network servers and security programs to office applications and development tools. Linux has evolved as part of the open source software movement in which independent programmers joined together to provide free, high-quality software to any user. Linux has become the premier platform for open source software, much of it developed by the Free Software Foundation's GNU project. Many of these applications are bundled as part of standard Linux distributions. Currently, thousands of open source applications are available for Linux from sites like SourceForge, Inc.'s sourceforge.net, K Desktop Environment's (KDE's) kde-apps.org and GNU Network Object Model Environment's (GNOME's) gnomefiles.org. Most of these applications are also incorporated into the distribution repository, using packages that are distribution compliant.

Along with Linux's operating system capabilities come powerful networking features, including support for Internet, intranets, and Windows networking. As a norm, Linux distributions include fast, efficient, and stable Internet servers, such as the web, File Transfer Protocol (FTP) and DNS servers, along with proxy, news, and mail servers. In other words, Linux has everything you need to set up, support, and maintain a fully functional network.

With both GNOME and KDE, Linux also provides GUIs with that same level of flexibility and power. Unlike Windows and the Mac, Linux enables you to choose the interface you want and then customize it further, adding panels, applets, virtual desktops, and menus, all with full drag-and-drop capabilities and Internet-aware tools.

Linux does all this at the right price. Linux is free, including the network servers and GUI desktops. Unlike the official Unix operating system, Linux is distributed freely under a GNU general public license as specified by the Free Software Foundation, making it available to anyone who wants to use it. GNU (the acronym stands for "GNUs Not Unix") is a project initiated and managed by the Free Software Foundation to provide free s/w to users, programmers and developers. Linux is copyrighted, not public domain. However a GNU public license has much the same effect as the software's being in the public domain. The GNU GPL is designed to ensure Linux remains free and, at the same time, standardized. Linux is technically the operating system kernel the core operations and only one official Linux kernel exists. People sometimes have the mistaken impression that Linux is somehow less than a professional operating system because it is free. Linux is in fact a PC, workstation and server version of UNIX. Many consider it far more stable and much more powerful than Windows. This power and stability have made Linux an operating system of choice as a network server.

To appreciate Linux completely, you need to understand the special context in which the UNIX operating system was developed. UNIX unlike most other operating systems, was developed in a research and academic environment. In universities, research laboratories, data centers and enterprises. Unix is the system most often used. Its development has paralleled the entire computer and communications revolution over the past several decades. Computer professionals often developed new computer technologies on Unix, such as those developed for the Internet. Although a sophisticated system, Unix was designed from the beginning to be flexible. The Unix system itself can be easily modified to create different versions. In fact many different vendors maintain different official versions of Unix, IBM, Sun and Hewlett- Packard all sell and maintain their own versions of Unix. The unique demands of research programs often require that Unix be tailored to their own special needs. This inherent flexibility in the Unix design in no way detracts from its quality. In fact, this flexibility attests to the ruggedness of Unix, allowing it to adapt to practically any environment. This is the context in which Linux was developed. Linux is, in this sense, one other version of Unix a version for the PC. The development of Linux by computer professionals working in a research like environment reflects the way Unix versions have usually been developed. Linux is publicly licensed and free—and reflects the deep roots Unix has in academic institutions, with their sense of public service and support. Linux is a top-rate operating system accessible to everyone, free of charge.

1.1 Linux Distributions

Although there is only one standard version of Linux, there are actually several different distributions. Different companies and groups have packaged Linux and Linux software in slightly different ways. Each company/group then releases the Linux package, usually on a CD-ROM. Later releases may include updated versions of programs or new software. Some of the more popular distributions are Red Hat, Ubuntu, Mepis, SUSE, Fedora and Debian. The Linux kernel is centrally distributed through kernel.org. All distributions use this same kernel, although it may be configured differently.

Linux has spawned a great variety of distributions. Many aim to provide a comprehensive solution providing support for any and all task. These include distributions like SUSE, Red Hat, and Ubuntu. Some are variations on other distributions, like Centos, which is based on Red Hat Enterprise Linux, and Ubuntu, which derives from Debian Linux. Others have been developed for more specialized tasks or to support certain features. Distributions like Debian provide cutting edge developments. Some distributions provide more commercial versions, usually bundled with commercial applications such as databases or secure servers. Certain companies like Red Hat and Novell provide a commercial distribution that corresponds to a supported free distribution. The free distribution is used to develop new features, like the Fedora Project for Red Hat. Other distributions like Knoppix and Ubuntu specialize in Live-CDs, the entire Linux operating system on single CD. Currently, distrowatch.com lists numerous Linux distributions. Check this site for details about current distributions. Table 1-1 lists the websites for several of the more popular Linux distributions. The FTP sites for these distributions use the prefix ftp instead of www, as in ftp.redhat.com. Also listed in Table 1-1 is the Linux kernel site where the newest releases of the official Linux kernel are provided. These sites have corresponding FTP sites where you can download updates and new releases.

1. 2 Operating Systems and Linux

An operating system is a program that manages computer hardware and software for the user. Operating systems were originally designed to perform repetitive hardware tasks, which centered around managing files, running programs, and receiving commands from the user. You interact with an operating system through a user interface, which allows the operating system to receive and interpret instructions sent by the user. You need only send an instruction to the operating system to perform a task, such as reading a file or printing a document. An operating system's user interface can be as simple as entering commands on a line or as complex as selecting menus and icons on a desktop.

An operating system also manages software applications. To perform different tasks such as editing documents or performing calculations you need specific software applications. An editor is an example of a software application that enables you to edit a document, making changes and adding new text. The editor itself is a program consisting of instructions to be executed by the computer. For the program to be used, it must first be loaded into computer memory, and then its instructions are executed. The operating system controls the loading and execution of all programs, including any software applications. When you want to use an editor, simply instruct the operating system to load the editor application and execute it.

File management, program management, and user interaction are traditional features common to all operating systems. Linux, like all versions of Unix, adds two more features. Linux is a multiuser and multitasking system. As it is a multitasking system, you can ask the system to perform several tasks at the same time. While one task is being done, you can work on another. For example, you can edit a file while another file is being printed. You do not have to wait for the other file to finish printing before you edit. As it is a multiuser system, several users can log in to the system at the same time, each interacting with the system through his or her own terminal.

As a version of Unix, Linux shares that system's flexibility, a flexibility stemming from Unix's research origins. Developed by Ken Thompson at AT&T Bell Laboratories in the late 1960s and early 1970s, the Unix system incorporated many new developments in operating system design. Originally Unix was designed as an operating system for researchers. One major goal was to create a system that could support the researchers' changing demands. To do this, Thompson had to design a system that could deal with many different kinds of tasks. Flexibility became more important than hardware efficiency. Like Unix, Linux has the advantage of being able to deal with the variety of tasks any user may face. The user is not confined to limited and rigid interactions with the operating system. Instead, the operating system is thought of as making a set of highly effective tools available to the user. This user- oriented philosophy means you can configure and program the system to meet your specific needs. With Linux, the operating system becomes an operating environment.

1.3 History of Unix and Linux

As a version of Unix, the history of Linux naturally begins with Unix. The story begins in the late 1960s, when a concerted effort to develop new operating system techniques occurred. In 1968, a consortium of researchers from General Electric, AT&T Bell Laboratories, and the Massachusetts Institute of Technology carried out a special operating system research project called MULTICS (the Multiplexed Information and Computing Service). MULTICS incorporated many new concepts in multitasking, file management, and user interaction.

Unix

In 1969, Ken Thompson, Dennis Ritchie, and the researchers at AT&T Bell Laboratories developed the Unix operating system incorporating many of the features of the MULTICS research project. They tailored the system for the needs of a research environment, designing it to run on minicomputers. From its inception, Unix was an affordable and efficient multiuser and multitasking operating system.

The Unix system became popular at Bell Labs as more and more researchers started using the system. In 1973, Dennis Ritchie collaborated with Ken Thompson to rewrite the programming code for the Unix system in the C programming language. Unix gradually grew from one person's tailored design to a standard software product distributed by many different vendors, such as Novell and IBM. Initially, Unix was treated as a research product. The first versions of Unix were distributed free to the computer science departments of many noted universities. Throughout the 1970s, Bell Labs began issuing official versions of Unix and licensing the systems to different users. One of these users was the computer science department of the University of California, Berkeley. Berkeley added many new features to the system that later became standard. In 1975 Berkeley released its own version of Unix, known by its distribution arm, Berkeley Software Distribution (BSD). This BSD version of Unix became a major contender to the AT&T Bell Labs version. AT&T developed several research versions of Unix, and in 1983 it released the first commercial version, called System 3. This was later followed by System V, which became a supported commercial software product.

At the same time, the BSD version of Unix was developing through several releases. In the late 1970s, BSD Unix became the basis of a research project by the Department of Defense's Advanced Research Projects Agency (DARPA). As a result, in 1983, Berkeley released a powerful version of Unix called BSD release 4.2. This release included sophisticated file management as well as networking features based on Internet network protocols—the same protocols now used for the Internet. BSD release 4.2 was widely distributed and adopted by many vendors, such as Sun Microsystems.

In the mid-1980s, two competing standards emerged, one based on the AT&T version of Unix and the other based on the BSD version. AT&T's Unix System Laboratories developed System V release 4. Several other companies, such as IBM and Hewlett-Packard, established the Open Software Foundation (OSF) to create their own standard version of Unix. Two commercial standard versions of Unix existed then—the OSF version and System V release 4.

Linux

Originally designed specifically for Intel-based PCs, Linux started out at the University of Helsinki as a personal project of a computer science student named Linus Torvalds. At that time, students were making use of a program called Minix, which highlighted different Unix features. Minix was created by Professor Andrew Tanenbaum and widely distributed over the Internet to students around the world. Linus's intention was to create an effective PC version of Unix for Minix users. It was named Linux, and in 1991, Linus released version 0.11. Linux was widely distributed over the Internet, and in the following years, other programmers refined and added to it, incorporating most of the applications and features now found in standard Unix systems. All the major window managers have been ported to Linux. Linux has all the networking tools, such as FTP support, web browsers, and the whole range of network services such as email, the domain name service, and dynamic host configuration, along with FTP, web, and print servers. It also has a full set of program development utilities, such as C++ compilers and debuggers. Given all its features, the Linux operating system remains small, stable, and fast. In its simplest format, Linux can run effectively on only 2MB of memory.

Although Linux has developed in the free and open environment of the Internet, it adheres to official Unix standards. Because of the proliferation of Unix versions in the previous decades, the Institute of Electrical and Electronics Engineers (IEEE) developed an independent Unix standard for the American National Standards Institute (ANSI). This new ANSI-standard Unix is called the Portable Operating System Interface for Computer Environments (POSIX). The standard defines how a Unix-like system needs to operate, specifying details such as system calls and interfaces.

POSIX defines a universal standard to which all Unix versions must adhere. Most popular versions of Unix are now POSIX-compliant. Linux was developed from the beginning according to the POSIX standard. Linux also adheres to the Linux file system hierarchy standard (FHS), which specifies the location of files and directories in the Linux file structure. See pathname.com/fhs for more details. Linux development is now overseen by The Linux Foundation (linux-found[illegible]n.org), which is a merger of The Free Standards Group and Open Source Development Labs (OS[illegible]). This is the group that Linus Torvalds works with to develop new Linux versions. Actual Linux kernels are released at kernel.org.

Linux Overview

Like Unix, Linux can be generally divided into three major components: the kernel, the environment, and the file structure. The kernel is the core program that runs programs and manages hardware devices, such as disks and printers. The environment provides an interface for the user. It receives commands from the user and sends those commands to the kernel for execution. The file structure organizes the way files are stored on a storage device, such as a disk. Files are organized into directories. Each directory may contain any number of subdirectories, each holding files. Together, the kernel, the environment, and the file structure form the basic operating system structure. With these three, you can run programs, manage files, and interact with the system.

An environment provides an interface between the kernel and the user. It can be described as an interpreter. Such an interface interprets commands entered by the user and sends them to the kernel. Linux provides several kinds of environments: desktops, window managers, and command line shells. Each user on a Linux system has his or her own user interface. Users can tailor their environments to their own special needs, whether they be shells, window managers, or desktops. In this sense, for the user, the operating system functions more as an operating environment, which the user can control.

In Linux, files are organized into directories, much as they are in Windows. The entire Linux file system is one large interconnected set of directories, each containing files. Some directories are standard directories reserved for system use. You can create your own directories for your own files, as well as easily move files from one directory to another. You can even move entire directories and share directories and files with other users on your system. With Linux, you can also set permissions on directories and files, allowing others to access them or restricting access to yourself alone. The directories of each user are, in fact, ultimately connected to the directories of other users. Directories are organized into a hierarchical tree structure, beginning with an initial root directory. All other directories are ultimately derived from this first root directory.

With KDE and GNOME, Linux now has a completely integrated GUI. You can perform all your Linux operations entirely from either interface. KDE and GNOME are fully operational desktops supporting drag-and-drop operations, enabling you to drag icons to your desktop and to set up your own menus on an Applications panel. Both rely on an underlying X Window System, which means as long as they are both installed on your system, applications from one can run on the other desktop. The GNOME and KDE sites are particularly helpful for documentation, news, and software you can download for those desktops. Both desktops can run any X Window System program, as well as any cursor-based program such as Emacs and Vi, which were designed to work in a shell environment. At the same time, a great many applications are written just for those desktops and included with your distributions. KDE and GNOME have complete sets of Internet tools, along with editors and graphics, multimedia, and system applications. Check their websites at gnome.org and kde.org for latest developments. As new versions are released, they include new software.

1.4 Open Source Software

Linux was developed as a cooperative open source effort over the Internet, so no company or institution controls Linux. Software developed for Linux reflects this background. Development often takes place when Linux users decide to work on a project together. The software is posted at an Internet site, and any Linux user can then access the site and download the software. Linux software development has always operated in an Internet environment and is global in scope, enlisting programmers from around the world. The only thing you need to start a Linux-based software project is a website.

Most Linux software is developed as open source software. This means that the source code for an application is freely distributed along with the application. Programmers over the Internet can make their own contributions to a software package's development, modifying and correcting the source code. Linux is an open source operating system as well. Its source code is included in all its distributions and is freely available on the Internet. Many major software development efforts are also open source projects, as are the KDE and GNOME desktops, along with most of their applications. The Netscape Communicator web browser package has also become open source, with its source code freely available. The OpenOffice office suite supported by Sun is an open source project based on the StarOffice office suite (StarOffice is essentially Sun's commercial version of OpenOffice). Many of the open source applications that run on Linux have located their websites at SourceForge (sourceforge.net), which is a hosting site designed specifically to support open source projects. You can find more information about the open source movement at opensource.org.

Open source software is protected by public licenses. These prevent commercial companies from taking control of open source software by adding a few modifications of their own, copyrighting those changes, and selling the software as their own product. The most popular public license is the GNU GPL provided by the Free Software Foundation. This is the license that Linux is distributed under. The GNU GPL retains the copyright, freely licensing the software with the requirement that the software and any modifications made to it always be freely available. Other public licenses have also been created to support the demands of different kinds of open source projects. The GNU lesser general public license (LGPL) lets commercial applications use GNU licensed software libraries. The qt public license (QPL) lets open source developers use the Qt libraries essential to the KDE desktop. You can find a complete listing at opensource.org.

Linux is currently copyrighted under a GNU public license provided by the Free Software Foundation, and it is often referred to as GNU software (see gnu.org). GNU software is distributed free, provided it is freely distributed to others. GNU software has proved both reliable and effective. Many of the popular Linux utilities, such as C compilers, shells, and editors, are GNU software applications. Installed with your Linux distribution are the GNU C++ and Lisp compilers, Vi and Emacs editors, BASH and TCSH shells, as well as TeX and Ghostscript document formatters. In addition, there are many open source software projects that are licensed under the GNU GPL.

Under the terms of the GNU GPL, the original author retains the copyright, although anyone can modify the software and redistribute it, provided the source code is included, made public, and provided free. Also, no restriction exists on selling the software or giving it away free. One distributor could charge for the software, while another one could provide it free of charge. Major software companies are also providing Linux versions of their most popular applications. Oracle provides a Linux version of its Oracle database. (At present, no plans seem in the works for Microsoft applications.)

Linux Software

All Linux software is currently available from online repositories. You can download applications for desktops, Internet servers, office suites, and programming packages, among others. Software packages may be distributed through online repositories. Downloads and updates are handled automatically by your desktop software manager and updater.
In addition, you can download from third-party sources software that is in the form of compressed archives or software packages like RPM and DEB. RPM packages are those archived using the Red Hat Package Manager, which is used on several distributions. Compressed archives have an extension such as .tar.gz or .tar.Z, whereas RPM packages have an .rpm extension and DEB uses a .deb extension. Any RPM package that you download directly, from whatever site, can be installed easily with the click of a button using a distribution software manager on a desktop. You can also download the source version and compile it directly on your system. This has become a simple process, almost as simple as installing the compiled RPM versions.Linux distributions also have a large number of mirror sites from which you can download their software packages for current releases. If you have trouble connecting to a main FTP site, try one of its mirrors.

Linux Office and Database Software

Many professional-level databases and office suites are now available for Linux. These include Oracle and IBM databases, as well as the OpenOffice and KOffice suites. Table 1-3 lists sites for office suites and databases. Most of the office suites, as well as MySQL and PostgreSQL, are already included on the distribution repositories and may be part of your install disk. Many of the other sites provide free personal versions of their software for Linux, and others are entirely free. You can download from them directly and install the software on your Linux system.

Internet Servers

One of the most important features of Linux, as of all Unix systems, is its set of Internet clients and servers. The Internet was designed and developed on Unix systems, and Internet clients and servers, such as those for FTP and the Web, were first implemented on BSD versions of Unix. DARPANET, the precursor to the Internet, was set up to link Unix systems at different universities across the nation. Linux contains a full set of Internet clients and servers, including mail, news, FTP, and web, as well as proxy clients and servers.

Development Resources

Linux has always provided strong support for programming languages and tools. All distributions include the GNU C and C++ (gcc) compiler with supporting tools such as make. Linux distributions usually come with full development support for the KDE and GNOME desktops, letting you create your own GNOME and KDE applications. You can also download the Linux version of the Java Software Development Kit for creating Java programs. A version of Perl for Linux is also included with most distributions. You can download current versions from their websites. Table 1-5 lists different sites of interest for Linux programming.

Online Linux Information Sources

Extensive online resources are available on almost any Linux topic. The tables in this chapter list sites where you can obtain software, display documentation, and read articles on the latest developments. Many Linux websites provide news, articles, and information about Linux. Several, such as linuxjournal.com, are based on popular Linux magazines. Some specialize in particular areas such as linuxgames.com for the latest games ported for Linux. Currently, many Linux websites provide news, information, and articles on Linux developments, as well as documentation, software links, and other resources.

Linux Documentation

Linux documentation has also been developed over the Internet. Much of the documentation currently available for Linux can be downloaded from Internet FTP sites. A special Linux project called the Linux Documentation Project (LDP), headed by Matt Welsh, has developed a complete set of Linux manuals. The documentation is available at the LDP home site, tldp.org. Linux documents provided by the LDP are listed in Table 1-7, along with their Internet sites. The Linux documentation for your installed software will be available at your /usr/share/doc directory.
An extensive number of mirrors are maintained for the LDP. You can link to any of them through a variety of sources, such as the LDP home site, tldp.org and linuxjournal.org. The documentation includes a user's guide, an introduction, and administrative guides.These are available in text, PostScript, or web page format. You can also find briefer explanations in what are referred to as HOW-TO documents.
Distribution websites provide extensive Linux documentation and software. The gnomerg site holds documentation for the GNOME desktop, while kde.org holds documentation for the KDE desktop. The tables in this chapter list many of the available sites. You can find other sites through resource pages that hold links to other websites—for example, the Linux website on the World Wide Web at tldp.org/links.html
In addition to websites, Linux Usenet newsgroups are also available. Through your Internet connection, you can access Linux newsgroups to read the comments of other Linux users and to post messages of your own. Several Linux newsgroups exist, each beginning with comp.os.linux. One of particular interest to the beginner is comp.os.linux.help, where you can post questions.

1.5 File Systems

Although all the files in your Linux system are connected into one overall directory tree, parts of that tree may reside on different storage devices such as hard drives or CD-ROMs. Files on a particular storage device are organized into what is referred to as a file system. A file system is a formatted device, with its own tree of directories and files. Your Linux directory tree may encompass several file systems, each on different storage devices. On a hard drive with several partitions, you have a file system for each partition. The files themselves are organized into one seamless tree of directories, beginning from the root directory. For example, if you attach a CD-ROM to your system, a pathname will lead directly from the root directory on your hard disk partition's file system to the files in the CD-ROM file system.

File System Hierarchy Standard (FHS)

Linux organizes its files and directories into one overall interconnected tree, beginning from the root directory and extending down to system and user directories. The organization and layout for the system directories are determined by the file system hierarchy standard (FHS). The FHS provides a standardized layout that all Linux distributions should follow in setting up their system directories. For example, there must be an /etc directory to hold configuration files and a /dev directory for device files. You can find out more about FHS, including the official documentation, at pathname.com/fhs. Linux distributions, developers, and administrators all follow the FHS to provide a consistent organization to the Linux file system.

Linux uses a number of specifically named directories for specialized administrative tasks. All these directories are at the very top level of your main Linux file system, the file system root directory, represented by a single slash, /. For example, the /dev directory holds device files, and the /home directory holds the user home directories and all their user files. You have access to these directories and files only as the system administrator (though users normally have read-only access). You need to log in as the root user, placing yourself in a special root user administrative directory called /root. From here, you can access any directory on the Linux file system, both administrative and user.

Root Directory: /

The subdirectories held in the root directory, /, are listed in Table 29-1, along with other useful subdirectories. Directories that you may commonly access as an administrator are the /etc directory, which holds configuration files; the /dev directory, which holds dynamically generated device files; and the /var directory, which holds server data files for DNS, web, mail, and FTP servers, along with system logs and scheduled tasks. For managing different versions of the kernel, you may need to access the /boot and /lib/modules directories as well as /usr/src/linux. The /boot directory holds the kernel image files for any new kernels you install, and the /lib/modules directory holds modules for your different kernels.

Directory	Function
/	Begins the file system structure—called the root.
/boot	Holds the kernel image files and associated boot information and files.
/home	Contains users' home directories.
/sbin	Holds administration-level commands and any commands used by the root user.
/dev	Holds dynamically generated file inter faces for devices such as the terminal and the printer (see "udev: Device Files" in Chapter 31).
/etc	Holds system configuration files and any other system files.
/etc/opt	Holds system configuration files for applications in /opt.
/etc/X11	Holds system configuration files for the X Window System and its applications.
/bin	Holds the essential user commands and utility programs.
/lib	Holds essential shared libraries and kernel modules.

/lib/modules	Holds the kernel modules.
/media	Holds directories for mounting media-based removable file systems, such as CD-ROMs, floppy disks, USB card readers, and digital cameras.
/mnt	Holds directories for additional file systems such as hard disks.
/opt	Holds added software applications (for example, KDE on some distributions).
/proc	Process director y, a memor y-resident director y that contains files used to provide information about the system.
/sys	Holds the sysfs file system for kernel objects, listing suppor ted kernel devices and modules.
/tmp	Holds temporar y files.
/usr	Holds those files and commands used by the system; this director y breaks down into several subdirectories.
/var	Holds files that var y, such as mailbox, web, and FTP files.

System Directories

Your Linux directory tree contains certain directories whose files are used for different system functions. For basic system administration, you should be familiar with the system program directories where applications are kept, the system configuration directory (/etc) where most configuration files are placed, and the system log directory (/var/log) that holds the system logs, recording activity on your system.

Directory	Description
/bin	Holds system-related programs.
/sbin	Holds system programs for specialized tasks.
/lib	Holds system libraries.
/etc	Holds configuration files for system and network ser vices and applications.

/home	Holds user home directories and ser ver data directories, such as website and FTP site files.
/media	Where removable media file systems like CD-ROMs, USB drives, and floppy disks are mounted.
/var	Holds system directories whose files continually change, such as logs, printer spool files, and lock files.
/usr	Holds user-related programs and files. Includes several key subdirectories, such as /usr/bin, /usr/X11, and /usr/share/doc
/usr/bin	Holds programs for users.
/dev	Holds device files.
/sys	Holds the sysfs file system with device information for kernel-supported devices on your system.
/usr/X11	Holds X Window System configuration files.
/usr/share	Holds shared files.
/usr/share/doc	Holds documentation for applications.
/usr/share/hal	Holds configuration for HAL removable devices.
/etc/udev	Holds configuration for device files.
/tmp	Holds system temporar y files.

Program Directories

Directories with bin in the name are used to hold programs. The /bin directory holds basic user programs, such as login, shells (BASH, TCSH, and zsh), and file commands (cp, mv, rm, ln, and so on). The /sbin directory holds specialized system programs for such tasks as file system management (fsck, fdisk, mkfs) and system operations like shutdown and startup (init). The /usr/bin directory holds program files designed for user tasks. The /usr/sbin directory holds user-related system operation, such as useradd for adding new users. The /lib directory holds all the libraries your system makes use of, including the main Linux library, libc, and subdirectories such as modules, which holds all the current kernel modules.

Configuration Directories and Files

When you configure different elements of your system, such as user accounts, applications, servers, or network connections, you make use of configuration files kept in certain system directories. Configuration files are placed in the /etc directory.

The /usr Directory

The /usr directory contains a multitude of important subdirectories used to support users, providing applications, libraries, and documentation. The /usr/bin directory holds numerous user-accessible applications and utilities; /usr/sbin hold user-accessible administrative utilities. The /usr/share directory holds architecture-independent data that includes an extensive number of subdirectories, including those for documentation, such as man, info, and doc files. Table 29-3 lists the subdirectories of the /usr directory.

The /media Directory

The /media directory is used for mountpoints for removable media like CD-ROM, DVD, floppy, or Zip drives, as well as for other media-based file systems such as USB card readers, cameras, and MP3 players. These are file systems you may be changing frequently, unlike partitions on fixed disks. Most Linux systems use the Hardware Abstraction Layer (HAL) to dynamically manage the creation, mounting, and device assignment of these devices. As instructed by HAL, this tool will create floppy, CD-ROM, storage card, camera, and MP3 player subdirectories in /media as needed. The default subdirectory for mounting is /media/disk. Additional drives have an number attached to their name.

The /mnt Directory

The /mnt directory is usually used for mountpoints for other mounted file systems such as Windows partitions. You can create directories for any partitions you want to mount, such as /mnt/windows for a Windows partition.

Directory	Description
/usr/bin	Holds most user commands and utility programs.
/usr/sbin	Holds administrative applications.
/usr/lib	Holds libraries for applications, programming languages, desktops, and so on.
/usr/games	Holds games and educational programs.
/usr/include	Holds C programming language header files (.h).
/usr/doc	Holds Linux documentation.
/usr/local	Holds locally installed software.
/usr/share	Holds architecture-independent data such as documentation.
/usr/src	Holds source code, including the kernel source code.
/usr/X11R6	Holds X Window System–based applications and libraries.

The /home Directory

The /home directory holds user home directories. When a user account is set up, a home directory is set up here for that account, usually with the same name as the user. As the system administrator, you can access any user's home directory, so you have control over that user's files.

The /var Directory

The /var directory holds subdirectories for tasks whose files change frequently, such as lock files, log files, web server files, or printer spool files. For example, the /var directory holds server data directories, such as /var/www for the Apache web server website files or /var/ftp for your FTP site files, as well as /var/named for the DNS server. The /tmp directory is simply a directory to hold any temporary files programs may need to perform a particular task.

The /var directories are designed to hold data that changes with the normal operation of the Linux system. For example, spool files for documents that you are printing are kept here. A spool file is created as a temporary printing file and is removed after printing. Other files, such as system log files, are changed constantly.

The /proc File System

The /proc file system is a special file system that is generated in system memory. It does not exist on any disk. /proc contains files that provide important information about the state of your system. For example, /proc/cpuinfo holds information about your computer’s CPU processor, /proc/devices lists those devices currently configured to run with your kernel, /proc/filesystems lists the file systems, and /proc files are really interfaces to the kernel, obtaining information from the kernel about your system. Table 29-5 lists the /proc subdirectories and files.Like any file system, /proc has to be mounted. The /etc/fstab file will have a special entry for /proc with a file system type of proc and no device specified.none/proc   proc   defaults 0   0

The sysfs File System: /sys

The sysfs file system is a virtual file system that provides the a hierarchical map of your kernel-supported devices such as PCI devices, buses, and block devices, as well as supporting kernel modules. The classes subdirectory will list all your supported devices by category, such as net and sound devices. With sysfs your system can easily determine the device file a particular device is associated with. This is very helpful for managing removable devices as well as dynamically configuring and managing devices as HAL and udev do. The sysfs file system is used by udev to dynamically generate needed device files in the /dev directory, as well as by HAL to manage removable device files and support as needed (HAL technically provides information only about devices, though it can use tools to dynamically change configurations as needed). The /sys file system type is sysfs. The /sys subdirectories organize your devices into different categories. The file system is used by systool to display a listing of your installed devices. The following example will list all your system devices Systool

Like /proc, the /sys directory resides only in memory, but you still need to mount it in the /etc/fstab file.

```
none    /sys     sysfs    defaults          0         0
```

Device Files: /dev, udev, and HAL

To mount a file system, you have to specify its device name. The interfaces to devices that may be attached to your system are provided by special files known as device files. The names of these device files are the device names. Device files are located in the /dev directories and usually have abbreviated names ending with the number of the device. For example, fd0 may reference the first floppy drive attached to your system. The prefix sd references SCSI hard drives, so sda2 would reference the second partition on the first SCSI hard drive. In most cases, you can use the man command with a prefix to obtain more detailed information about this kind of device. For example, man sd displays the Man pages for SCSI devices. A complete listing of all device names can be found in the devices file located in the linux/doc/device-list directory at the kernel.org website and in the devices.txt file in the /etc/usr/linux-2.4/Documentation directory on your system. Table 29-6 lists several of the commonly used device names.

udev and HAL

Device files are no longer handled in a static way; they are now dynamically generated as needed instead. Previously a device file was created for each possible device, leading to a very large number of device files in the /etc/dev directory. Now, your system will detect only those devices it uses and create device files for those only, giving you a much smaller listing of device files. The tool used to detect and generate device files is udev, user devices. Each time your system is booted, udev will automatically detect your devices and generate device files for them in the /etc/dev directory. This means that the /etc/dev directory and its files are recreated each time you boot.

It is a dynamic directory, no longer static. To manage these device files, you need to use udev configuration files located in the /etc/udev directory. This means that udev is able to also dynamically manage all removable devices; udev will generate and configure devices files for removable devices as they are attached, and then remove these files when the devices are removed. In this sense, all devices are now considered hotplugged, with fixed devices simply being hotplugged devices that are never removed.

As /etc/dev is now dynamic, any changes you would make manually to the /etc/dev directory will be lost when you reboot. This includes the creation of any symbolic links such as /dev/cdrom that many software applications use. Instead, such symbolic links have to be configured using udev rules listed in configuration files located in the /etc/udev/rules.d directory. Default rules are already in place for symbolic links, but you can create rules of your own.

In addition to udev, information about removable devices like CD-ROMs and floppy disks, along with cameras and USB printers, used by applications like the desktop to dynamically interface with them, is managed by a separate utility called the Hardware Abstract Layer (HAL). HAL allows a removable device like a USB printer to be recognized no matter what particular connections it may be using. For example, you can attach a USB printer in one USB port at one time and then switch it to another later. The fstab file is edited using the fstab-sync tool, which is invoked by HAL rules in configuration files in/usr/share/hal/fdi directory.

HAL has a key impact on the /etc/fstab file used to manage file systems. No longer are entries maintained in the /etc/fstab file for removable devices like your CD-ROMs. These devices are managed directly by HAL using its set of storage callouts like hal-system- storage-mount to mount a device or hal-system-storage-eject to remove one. In effect you now have to use the HAL device information files to manage your removable file systems. Should you want to bypass HAL and manually configure a CD-ROM device, you simply place an entry for it in the /etc/fstab file.

Floppy and Hard Disk Devices

The device name for your floppy drive is fd0; it is located in the directory /dev. /dev/fd0 references your floppy drive. Notice the numeral 0 after fd. If you have more than one floppy drive, additional drives are represented by fd1, fd2, and so on.IDE hard drives use the prefix hd, whereas SCSI hard drives use the prefix sd. RAID devices, on the other hand, use the prefix md. The prefix for a hard disk is followed by a letter that labels the hard drive and a number for the partition. For example, hda2 references the second partition on the first IDE hard drive, where the first hard drive is referenced with the letter a, as in hda. The device sdb3 refers to the third partition on the second SCSI hard drive (sdb). RAID devices, however, are numbered from 0, like floppy drives. Device md0 references the first RAID device, and md1 references the second. On an IDE hard disk device, Linux supports up to four primary IDE hard disk partitions, numbered 1 through 4. You are allowed any number of logical partitions. To find the device name, you can use df to display your hard partitions or examine the /etc/fstab file.

CD-ROM Devices

The device name for your CD-ROM drive varies depending on the type of CD-ROM you have. The device name for an IDE CD-ROM has the same prefix as an IDE hard disk partition, hd, and is identified by a following letter that distinguishes it from other IDE devices. For example, an IDE CD-ROM connected to your secondary IDE port may have the name hdc. An IDE CD-ROM connected as a slave to the secondary port may have the name hdd. The actual name is determined when the CD-ROM is installed, as happened when you installed your Linux system. SCSI CD-ROM drives use a different nomenclature for their device names. They begin with scd for SCSI drive and are followed by a distinguishing number. For example, the name of a SCSI CD-ROM could be scd0 or scd1. The name of your CD-ROM was determined when you installed your system.

As noted previously, CD-ROM devices are now configured by HAL. HAL does this in a device information file in its policy configuration directory. To configure a CD-ROM device, as by adding user mount capability, you need to configure its entry in the storage-methods .fdi configuration files (see Chapter 31 for details). The GNOME Volume Manager uses HAL and udev to access removable media directly and Samba to provide Windows networking support. Media are mounted by gnome-mount, a wrapper for accessing HAL and udev, which perform the mount (/etc/fstab is no longer used).

1.6 FILES, FILETYPES, FILE OWNERSHIP, AND FILE PERMISSIONS

Managing files under Linux is different from managing files under Windows NT/200x/ XP/Vista, and radically different from managing files under Windows 95/98. In this sec- tion, we discuss basic file management tools and concepts under Linux. We'll start with specifics on some useful general-purpose commands, and then we'll step back and look at some background information.

Under Linux (and UNIX in general), almost everything is abstracted to a file. Origi- nally, this was done to simplify the programmer's job. Instead of having to communicate directly with device drivers, special files (which look like ordinary files to the applica- tion) are used as a bridge. Several types of files accommodate all these file uses.

Normal Files

Normal files are just that—normal. They contain data or executables, and the operating system makes no assumptions about their contents.

Directories

Directory files are a special instance of normal files. Directory files list the locations of other files, some of which may be other directories. (This is similar to folders in Windows.) In general, the contents of directory files won't be of importance to your daily operations, unless you need to open and read the file yourself rather than using existing applications to navigate directories. (This would be similar to trying to read the DOS file allocation table directly rather than using command.com to navigate directories or using the find- first/findnext system calls.)

Hard Links

Each file in the Linux file system gets its own i-node. An i-node keeps track of a file's attributes and its location on the disk. If you need to be able to refer to a single file using two separate filenames, you can create a hard link. The hard link will have the same i-node as the original file and will, therefore, look and behave just like the original. With every hard link that is created, a reference count is incremented. When a hard link is removed, the reference count is decremented. Until the reference count reaches zero, the file will remain on disk.
Symbolic Links

Unlike hard links, which point to a file by its i-node, a symbolic link points to another file by its name. This allows symbolic links (often abbreviated symlinks) to point to files located on other partitions, even other network drives.

Block Devices

Since all device drivers are accessed through the file system, files of type block device are used to interface with devices such as disks. A block device file has three identify- ing traits:

It has a major number. It has a minor number.

- When viewed using the ls -l command, it shows b as the first character of the permissions field.

For example,

```
[yyang@fedora-serverA ~]$ ls -l /dev/sda
brw-r----- 1 root disk 8, 0 2090-09-30 08:18 /dev/sda
```

Note the b at the beginning of the file's permissions; the 8 is the major number, and the 0 is the minor number.

A block device file's major number identifies the represented device driver. When this file is accessed, the minor number is passed to the device driver as a parameter, telling it which device it is accessing. For example, if there are two serial ports, they will share the same device driver and thus the same major number, but each serial port will have a unique minor number.

Character Devices

Similar to block devices, character devices are special files that allow you to access devices through the file system. The obvious difference between block and character devices is that block devices communicate with the actual devices in large blocks, whereas char- acter devices work one character at a time. (A hard disk is a block device; a modem is a character device.) Character device permissions start with a c, and the file has a major number and a minor number. For example,

```
[yyang@fedora-serverA ~]$ ls -l /dev/ttyS0
crw-rw---- 1 root uucp 4, 64 2007-09-30 08:18 /dev/ttyS0
```

Named Pipes

Named pipes are a special type of file that allows for interprocess communication. Using the mknod command, you can create a named pipe file that one process can open for reading and another process can open for writing, thus allowing the two to communicate with one another. This works especially well when a program refuses to take input from a command-line pipe, but another program needs to feed the other one data and you don't have the disk space for a temporary file. For a named pipe file, the first character of its file permissions is a p. For example, if a named pipe called mypipe exists in your present working directory (PWD), a long listing of the named pipe file would show this:

```
[yyang@fedora-serverA ~]$ ls -l mypipe
prw-r--r--     1  root    root          0 Mar 16 10:47 mypipe
```

Listing Files: ls

Out of necessity, we have been using the ls command in previous sections and chapters of this book. We will look at the ls command and its options in more details here.

The ls command is used to list all the files in a directory. Of more than 50 available options, the ones listed in Table 5-2 are the most commonly used. The options can be used in any combination.

To list all files in a directory with a long listing, type this command:

```
[yyang@fedora-serverA ~]$ ls -la
```

To list a directory's nonhidden files that start with the letter A, type this:

```
[yyang@fedora-serverA ~]$ ls A*
```

Option for ls	Description
-l	Long listing. In addition to the filename, shows the file size, date/time, permissions, ownership, and group information.
-a	All files. Shows all files in the directory, including Names of hidden files begin with a period.
-t	Lists in order of last modified time.
-r	Reverses the listing.
-1	Single-column listing.
-R	Recursively lists all files and subdirectories.

Change Ownership: chown

The chown command allows you to change the ownership of a file to someone else. Only the root user can do this. (Normal users may not give away file ownership or steal own- ership from another user.) The syntax of the command is as follows:

```
[root@fedora-serverA ~]# chown [-R] username filename
```

where username is the login of the user to whom you want to assign ownership, and filename is the name of the file in question. The filename may be a directory as well.

The -R option applies when the specified filename is a directory name. This option tells the command to recursively descend through the directory tree and apply the new ownership, not only to the directory itself, but also to all of the files and directories within it.

Change Group: chgrp

The chgrp command-line utility lets you change the group settings of a file. It works much like chown. Here is the format:

```
[root@fedora-serverA ~]# chgrp [-R] groupname filename
```

where groupname is the name of the group to which you want to assign filename own- ership. The filename may be a directory as well.

The -R option applies when the specified filename is a directory name. As with chown, the -R option tells the command to recursively descend through the directory tree and apply the new ownership, not only to the directory itself, but also to all of the files and directories within it.

Change Mode: chmod

Directories and files within the Linux system have permissions associated with them. By default, permissions are set for the owner of the file, the group associated with the file, and everyone else who can access the file (also known as owner, group, and other, respectively). When you list files or directories, you see the permissions in the first col- umn of the output. Permissions are divided into four parts. The first part is represented by the first character of the permission. Normal files have no special value and are rep- resented with a hyphen (-) character. If the file has a special attribute, it is represented by a letter. The two special attributes we are most interested in here are directories (d) and symbolic links (l).

The second, third, and fourth parts of a permission are represented in three-character chunks. The first part indicates the file owner's permission. The second part indicates the group permission. The last part indicates the world permission. In the context of UNIX, "world" means all users in the system, regardless of their group settings.
Following are the letters used to represent permissions and their corresponding values. When you combine attributes, you add their values. The chmod command is used to set permission values.

Letter	Permission	Value
r	Read	4
w	Write	2
x	Execute	1

Using the numeric command mode is typically known as the octal permissions, since the value can range from 0–7. To change permissions on a file, you simply add these values together for each permission you want to apply.
For example, if you want to make it so that just the user (owner) can have full access (RWX) to a file called foo, you would type

```
[yyang@fedora-serverA ~]$ chmod 700 foo
```

What is important to note is that using the octal mode, you always replace any permis- sions that were set. So if there was a file in /usr/local that was SetUID and you ran the command chmod -R 700 /usr/local, that file will no longer be SetUID. If you want to change certain bits, you should use the symbolic mode of chmod. This mode turns out to be much easier to remember, and you can add, subtract, or overwrite permissions.
The symbolic form of chmod allows you to set the bits of the owner, the group, or others. You can also set the bits for all. For example, if you want to change a file called foobar.sh so that it is executable for the owner, you can run the following command:

```
[yyang@fedora-serverA ~]$ chmod u+x foobar.sh
```

If you want to change the group's bit to execute also, use the following:

```
[yyang@fedora-serverA ~]$ chmod ug+x foobar.sh
```

If you need to specify different permissions for others, just add a comma and its per- mission symbols, as here:

```
[yyang@fedora-serverA ~]$ chmod ug+x,o-rwx foobar.sh
```

If you do not want to add or subtract a permission bit, you can use the equal (=) sign instead of a plus (+) sign or minus (-) sign. This will write the specific bits to the file and erase any other bit for that permission. In the previous examples, we used + to add the execute bit to the User and Group fields. If you want only the execute bit, you would replace the + with =. There is also a fourth character you can use: a. This will apply the permission bits to all of the fields.
The following list shows the most common combinations of the three permissions. Other combinations, such as -wx, do exist, but they are rarely used.

Letter	Permission	Value
r--	Read only	4
rw-	Read and write	6
rwx	Read, write, and execute	7
r-x	Read and execute	5
--x	Execute only	1

for each file these three-letter chunks are grouped together. The first chunk represents the permissions for the owner of the file, the second chunk represents the per- missions for the file's group, and the last chunk represents the permissions for all users on the system. Table shows some permission combinations, their numeric equiva- lents, and their descriptions.

Permission	Numeric Equivalent	Description
		the mkdir command. Only the owner can read and write to this directory. Note that all directories must have the
drwxr-xr-x	755	This directory can be changed only by the owner but everyone else can view its contents.
drwx--x--x	711	A handy combination for keeping a directory world- readable but restricted from access by the ls command. A file can be read only by someone who knows the filename.

1.7 FILE MANAGEMENT AND MANIPULATION

This section covers the basic command-line tools for managing files and directories. Most of this will be familiar to anyone who has used a command-line interface—same old functions, but new commands to execute.

Copy Files: cp

The cp command is used to copy files. It has a substantial number of options. See its man page for additional details. By default, this command works silently, only displaying status information if an error condition occurs. Following are the most common options for cp:

Option for cp	Description
-f	Forces copy; does not ask for verification
-I	Interactive copy; before each file is copied, verifies with user

First, let's use the touch command to create an empty file called foo.txt in the user yyang's home directory. Type

```
[yyang@fedora-serverA ~]$ touch foo.txt
```

To use the cp (copy) command to copy foo.txt to foo.txt.html, type

```
[yyang@fedora-serverA ~]$ cp foo.txt foo.txt.html
```

To interactively copy all files ending in .html to the /tmp directory, type this command:

```
[yyang@fedora-serverA ~]$ cp -i *.html /tmp
```

Move Files: mv

The mv command is used to move files from one location to another. Files can be moved across partitions/file systems as well. Moving files across partitions involves a copy operation, and as a result, the move command may take longer. But you will find that moving files within the same file system is almost instantaneous. Following are the most common options for mv:

Option for mv	Description
-f	Forces move
-I	Interactive move

To move a file named foo.txt.html from /tmp to your present working directory, use this command:

```
[yyang@fedora-serverA ~]$ mv /tmp/foo.txt.html
```

Link Files: ln

The ln command lets you establish hard links and soft links (see "Files, File Types, File Ownership, and File Permissions" earlier in this chapter). The general format of ln is as follows:

```
[yyang@fedora-serverA ~]$ ln original_file new_file
```

Although ln has many options, you'll rarely need to use most of them. The most common option, -s, creates a symbolic link instead of a hard link.

To create a symbolic link called link-to-foo.txt that points to the original file called foo.txt, issue the command

```
[yyang@fedora-serverA ~]$ ln -s foo.txt link-to-foo.txt
```

Find a File: find

The find command lets you search for files according to various criteria. Like the tools we have already discussed, find has a large number of options that you can read about in its man page. Here is the general format of find:

```
[yyang@fedora-serverA ~]$ find start_dir [options]
```

where start_dir is the directory from which the search should start.

To find all files in the current directory (i.e., the "." directory) that have not been accessed in at least seven days, use the following command:

```
[yyang@fedora-serverA ~]$ find . -atime 7
```

Type this command to find all files in your present working directory whose names are core and then delete them (i.e., automatically run the rm command):

```
[yyang@fedora-serverA ~]$ find . -name core -exec rm {} \;
```

Create a Directory: mkdir

The mkdir command in Linux is identical to the same command in other flavors of Linux, as well as in MS-DOS. An often-used option of the mkdir command is the -p option. This option will force mkdir to create parent directories if they don't exist already. For example, if you need to create /tmp/bigdir/subdir/mydir and the only directory that exists is /tmp, using -p will cause bigdir and subdir to be automatically created along with mydir.

Create a directory tree like bigdir/subdir/finaldir in your PWD. Type

```
[yyang@fedora-serverA ~]$ mkdir  -p bigdir/subdir/finaldir
```

To create a single directory called mydir, use this command:

```
[yyang@fedora-serverA ~]$ mkdir mydir
```

Remove a Directory: rmdir

The rmdir command offers no surprises for those familiar with the DOS version of the command; it simply removes an existing directory. This command also accepts the -p parameter, which removes parent directories as well.

For example, if you want to get rid of all the directories from bigdir to finaldir that were created earlier, you'd issue this command alone:

```
[yyang@fedora-serverA ~]$ rmdir -p bigdir/subdir/finaldir
```

To remove a directory called mydir, you'd type this:

```
[yyang@fedora-serverA ~]$ rmdir mydir
```

Show Present Working Directory: pwd

It is inevitable that you will sit down in front of an already logged-in workstation and not know where you are in the directory tree. To get this information, you need the pwd command. Its only task is to print the current working directory. To display your current working directory, use this command:

```
[yyang@fedora-serverA ~]$pwd
/home/yyan
```

Concatenate Files: cat

The cat program fills an extremely simple role: to display files. More creative things can be done with it, but nearly all of its usage will be in the form of simply displaying the contents of text files

—much like the type command under DOS. Because multiple filenames can be specified on the command line, it's possible to concatenate files into a single, large, continuous file. This is different from tar in that the resulting file has no control information to show the boundaries of different files.

To display the /etc/passwd file, use this command:

```
[yyang@fedora-serverA ~]$ cat /etc/passwd
```

To display the /etc/passwd file and the /etc/group file, issue this command:

```
[yyang@fedora-serverA ~]$ cat /etc/passwd /etc/group
```

Type this command to concatenate /etc/passwd with /etc/group and send the output into the file users-and-groups.txt:

```
[yyang@fedora-serverA ~]$ cat /etc/passwd /etc/group > users-and-groups.txt
```

To append the contents of the file /etc/hosts to the users-and-groups.txt file you just created, type

```
[yyang@fedora-serverA ~]$ cat /etc/hosts >> users-and-groups.txt
```

Display a File One Screen at a Time: more

The more command works in much the same way the DOS version of the program does. It takes an input file and displays it one screen at a time. The input file can come either from its stdin or from a command-line parameter. Additional command-line param- eters, though rarely used, can be found in the man page.

To view the /etc/passwd file one screen at a time, use this command:

```
[yyang@fedora-serverA ~]$ more /etc/passwd
```

To view the directory listing generated by the ls command one screen at a time, enter

```
[yyang@fedora-serverA ~]$ ls | more
```

List Processes: ps

The ps command lists all the processes in a system, their state, size, name, owner, CPU time, wall clock time, and much more. Many command-line parameters are available; the ones most often used are described in Table

Option for ps	Description
-a	Shows all processes with a controlling terminal, not just the current user's processes
-r	Shows only running processes (see the description of process states later in this section)
-x	Shows processes that do not have a controlling terminal
-u	Shows the process owners
-f	Displays parent/child relationships among processes
-l	Produces a list in long format
-w	Shows a process's command-line parameters
-ww	Shows a process's command-line parameters (unlimited width fashion)

Send a Signal to a Process: kill

This program's name is misleading: It doesn't really kill processes. What it does is send signals to running processes. The operating system, by default, supplies each process with a standard set of signal handlers to deal with incoming signals. From a system administrator's standpoint, the most common handlers are for signals number 9 and 15, kill process and terminate process, respectively. When kill is invoked, it requires at least one parameter: the process identification number (PID) as derived from the ps command. When passed only the PID, kill sends signal 15. Some programs intercept this signal and perform a number of actions so that they can shut down cleanly. Others just stop running in their tracks. Either way, kill isn't a guaranteed method for making a process stop.

Who Is Logged In: who

On systems that allow users to log into other users' machines or special servers, you will want to know who is logged in. You can generate such a report by using the who command:

```
[yyang@fedora-serverA ~]$ who
        yyang   pts/0       2010-10-08 15:24 (10.35.35.51)

        yyang   pts/1       2010-10-08 16:07 (10.35.35.51)
```

1.8 THE MAKEUP OF FILE SYSTEMS

Let's begin by going over the structure of file systems under Linux. This will help to clarify your understanding of the concept and let you see more easily how to take advantage of the architecture.

i-Nodes

The most fundamental building block of many Linux/UNIX file systems is the i-node. An i-node is a control structure that points either to other i-nodes or to data blocks.

The control information in the i-node includes the file's owner, permissions, size, time of last access, creation time, group ID, and so on. (For the truly curious, the entire kernel data structure for the ext2 file system is available in /usr/src/kernels/*/include/ linux/ext2_fs.h—assuming, of course, that you have the source tree installed in the /usr/ src directory.) The one information an i-node does not provide is the file's name.

Directories themselves are special instances of files. This means each directory gets an i-node, and the i-node points to data blocks containing information (filenames and i-nodes) about the files in the directory. Figure 7-1 illustrates the organization of i-nodes and data blocks in the ext2 file system.

The i-nodes are used to provide indirection so that more data blocks can be pointed to—which is why each i-node does not contain the filename. (Only one i-node works as a representative for the entire file; thus, it would be a waste of space if every i-node contained filename information.) Take, for example, a 6-gigabyte (GB) disk that contains 1,079,304 i-nodes. If every i-node consumed 256 bytes to store the filename, a total of about 33 megabytes (MB) would be wasted in storing filenames, even if they weren't being used!

Superblocks

The first piece of information read from a disk is its superblock. This small data structure reveals several key pieces of information, including the disk's geometry, the amount of available space, and, most importantly, the location of the first i-node. Without a super- block, an on-disk file system is useless.

Something as important as the superblock is not left to chance. Multiple copies of this data structure are scattered all over the disk to provide backup in case the first one is damaged. Under Linux's ext2 file system, a superblock is placed after every group of blocks, which contains i-nodes and data. One group consists of 8192 blocks; thus, the first redundant superblock is at

8193, the second at 16,385, and so on. The designers of most Linux file systems intelligently included this superblock redundancy into the file system design.

ext3 and ReiserFS

Ext3 and ReiserFS are two popular Linux file systems used by the major Linux distributions. The ext3 file system is an enhanced extension of the ext2 file system. As of this writing, the ext2 file system is somewhere around 16 years old. This means two things for us as system administrators. First and foremost, ext2 is rock-solid. It is a well-tested subsystem of Linux and has had the time to become well optimized. Second, other file systems that were considered experimental when ext2 was created have matured and become available to Linux.

The two file systems that are popular replacements for ext2 are the ext3 and ReiserFS file systems. Both offer significant improvements in performance and stability, but the most important component of both is that they have moved to a new method of getting the data to the disk. This new method is called journaling. Traditional file systems (such as ext2) must search through the directory structure, find the right place on disk to lay out the data, and then lay out the data. (Linux can also cache the whole process, includ- ing the directory updates, thereby making the process appear faster to the user.) Almost all new versions of Linux distributions now make use of one journaling file system or the other by default. Fedora (and other Red Hat Enterprise Linux [RHEL] derivatives), OpenSuSE, and Ubuntu, for example, use ext3 by default.

The problem with not having a journaling file system is that in the event of an unex- pected crash, the file system checker or file system consistency checker (fsck) program has to follow up on all of the files on the disk to make sure they don't contain any dan- gling references (for example, i-nodes that point to other, invalid i-nodes or data blocks). As disks expand in size and shrink in price, the availability of these large-capacity disks means more of us will have to deal with the aftermath of having to fsck a large disk. And as anyone who has had to do that before can tell you, it isn't fun. The process can take a long time to complete, and that means downtime for your users.

Journaling file systems work by first creating an entry of sorts in a log (or journal) of changes that are about to be made before actually committing the changes to disk. Once this transaction has been committed to disk, the file system goes ahead and modifies the actual data or metadata. This results in an all-or-nothing situation; that is, either all or none of the file system changes get done.

One of the benefits of using a journaling-type file system is the greater assurance that data integrity will be preserved, and in the unavoidable situations where problems arise, speed, ease of recovery, and likelihood of success are vastly increased. One such unavoidable situation might be in the event of a system crash. Here, you may not need to run fsck. Think how much faster you could recover a system if you didn't have to run fsck on a 1 TB disk! (Haven't had to run fsck on a big disk before? Think about how long it takes to run ScanDisk under Windows on large disks.) Other benefits of using journaling-type file systems are that system reboots are simplified, disk fragmentation is reduced, and input/output (I/O) operations can be accelerated (this depends on the journaling method used).

1.10 Basic File Attributes

The UNIX file system allows the user to access other files not belonging to them and without infringing on security. A file has a number of attributes (properties) that are stored in the inode. In this chapter, we discuss,

ls –l to display file attributes (properties)
Listing of a specific directory
Ownership and group ownership
Different file permissions

Listing File Attributes

ls command is used to obtain a list of all filenames in the current directory. The output in UNIX lingo is often referred to as the listing. Sometimes we combine this option with other options for displaying other attributes, or ordering the list in a different sequence. ls look up the file’s inode to fetch its attributes. It lists seven attributes of all files in the current directory and they are:

File type and Permissions
Links
Ownership
Group ownership
File size
Last Modification date and time
File name

The file type and its permissions are associated with each file. Links indicate the number of file names maintained by the system. This does not mean that there are so many copies of the file. File is created by the owner. Every user is attached to a group owner. File size in bytes is displayed. Last modification time is the next field. If you change only the permissions or ownership of the file, the modification time remains unchanged. In the last field, it displays the file name.
For example,

```
$ ls –l

total 72
-rw-r--r--    1    kumar  metal  19514  may  10   13:45    chap01
-rw-r--r--    1    kumar  metal  4174   may  10   15:01    chap02
-rw-rw-rw-    1    kumar  metal  84     feb  12   12:30    dept.lst
-rw-r--r--    1    kumar  metal  9156   mar  12   1999     genie.sh
drwxr-xr-x    2    kumar  metal  512    may   9   10:31    helpdir
drwxr-xr-x    2    kumar  metal  512    may   9   09:57    progs
```

Listing Directory Attributes

ls -d will not list all subdirectories in the current directory For example,

```
ls –ld helpdir progs
drwxr-xr-x  2  kumar  metal   512  may   9   10:31 helpdir
drwxr-xr-x  2  kumar  metal   512  may   9   09:57 progs
```

Directories are easily identified in the listing by the first character of the first column, which here shows a d. The significance of the attributes of a directory differs a good deal from an ordinary file. To see the attributes of a directory rather than the files contained in it, use ls –ld with the directory name.

Note that simply using ls –d will not list all subdirectories in the current directory. Strange though it may seem, ls has no option to list only directories.

File Ownership

When you create a file, you become its owner. Every owner is attached to a group owner. Several users may belong to a single group, but the privileges of the group are set by the owner of the file and not by the group members. When the system administrator creates a user account, he has to assign these parameters to the user:
The user-id (UID) – both its name and numeric representation The group-id (GID) – both its name and numeric representation

File Permissions

UNIX follows a three-tiered file protection system that determines a file's access
rights. It is displayed in the following format:
Filetype owner (rwx) groupowner (rwx) others (rwx) For Example:
-rwxr-xr-- 1 kumar metal 20500 may 10 19:21 chap02 r w x r - x r - -
owner/user group owner others
The first group has all three permissions. The file is readable, writable and executable by the owner of the file. The second group has a hyphen in the middle slot, which indicates the absence of write permission by the group owner of the file. The third group has the write and execute bits absent. This set of permissions is applicable to others.
You can set different permissions for the three categories of users – owner, group and others. It's important that you understand them because a little learning here can be a dangerous thing. Faulty file permission is a sure recipe for disaster

Changing File Permissions

A file or a directory is created with a default set of permissions, which can be determined by umask. Let us assume that the file permission for the created file is -rw-r-- r--. Using chmod command, we can change the file permissions and allow the owner to execute his file. The command can be used in two ways:
In a relative manner by specifying the changes to the current permissions In an absolute manner by specifying the final permissions

Relative Permissions

chmod only changes the permissions specified in the command line and leaves the other permissions unchanged. Its syntax is:
chmod category operation permission filename(s) chmod takes an expression as its argument which contains: user category (user, group, others)

Category	operation	permission
u – user g - group o - others a - all (ugo)	+ assign - remove = absolute	r - read w - write x - execute

Let us discuss some examples: Initially,

```
-rw-r--r--        1     kumar    metal  1906  sep   23:38   xstart

chmod u+x xstart

-rwxr--r--        1     kumar    metal  1906  sep   23:38   xstart
```

The command assigns (+) execute (x) permission to the user (u), other permissions remain unchanged.
chmod ugo+x xstart or
chmod a+x xstart or
chmod +x xstart
-rwxr-xr-x 1 kumar metal 1906 sep 23:38 xstart chmod accepts multiple file names in command line
chmod u+x note note1 note3 Let initially,
-rwxr-xr-x 1 kumar metal 1906 sep 23:38 xstart chmod go-r xstart
Then, it becomes -rwx--x--x 1 kumar metal 1906 sep 23:38 xstart

Absolute Permissions

Here, we need not to know the current file permissions. We can set all nine permissions explicitly. A string of three octal digits is used as an expression. The permission can be represented by one octal digit for each category. For each category, we add octal digits. If we represent the permissions of each category by one octal digit, this is how the permission can be represented:
Read permission – 4 (octal 100)
Write permission – 2 (octal 010)
Execute permission – 1 (octal 001)

Octal	Permissions	Significance
0	- - -	no permissions
1	- - x	execute only
2	- w -	write only
3	- w x	write and execute
4	r - -	read only
5	r - x	read and execute
6	r w -	read and write
7	r w x	read, write and execute

We have three categories and three permissions for each category, so three octal digits can describe a file's permissions completely. The most significant digit represents user and the least one represents others. chmod can use this three-digit string as the expression.
Using relative permission, we have,
chmod a+rw xstart
Using absolute permission, we have, chmod 666 xstart

chmod 644 xstart

chmod 761 xstart
will assign all permissions to the owner, read and write permissions for the group and only execute permission to the others.

777 signify all permissions for all categories, but still we can prevent a file from being deleted. 000 signifies absence of all permissions for all categories, but still we can delete a file. It is the directory permissions that determine whether a file can be deleted or not. Only owner can change the file permissions. User can not change other user's file's permissions. But the system administrator can do anything.

The Security Implications

Let the default permission for the file xstart is
-rw-r--r--
chmod u-rw, go-r xstart   or
chmod 000 xstart

This is simply useless but still the user can delete this file On the other hand,
chmod a+rwx xstart chmod 777 xstart
-rwxrwxrwx
The UNIX system by default, never allows this situation as you can never have a secure system. Hence, directory permissions also play a very vital role here. We can use chmod Recursively.
chmod -R a+x shell_scripts

This makes all the files and subdirectories found in the shell_scripts directory, executable by all users. When you know the shell meta characters well, you will appreciate that the * doesn't match filenames beginning with a dot. The dot is generally a safer but note that both commands change the permissions of directories also.

Directory Permissions

It is possible that a file cannot be accessed even though it has read permission, and can be removed even when it is write protected. The default permissions of a directory are,
rwxr-xr-x (755)
A directory must never be writable by group and others Example:
mkdir c_progs ls –ld c_progs
drwxr-xr-x   2 kumar metal 512 may 9 09:57 c_progs
If a directory has write permission for group and others also, be assured that every user can remove every file in the directory. As a rule, you must not make directories universally writable unless you have definite reasons to do so.

Changing File Ownership

Usually, on BSD and AT&T systems, there are two commands meant to change the ownership of a file or directory. Let kumar be the owner and metal be the group owner. If sharma copies a file of kumar, then sharma will become its owner and he can manipulate the attributes
chown changing file owner and chgrp changing group owner On BSD, only system administrator can use chown. On other systems, only the owner can change both

chown

Changing ownership requires superuser permission, so use su command ls -l note

```
-rwxr----x      1 kumar  metal  347  may  10  20:30  note
chown sharma note; ls -l note
-rwxr----x      1 sharma  metal  347  may  10  20:30  note
```

Once ownership of the file has been given away to sharma, the user file permissions that previously applied to Kumar now apply to sharma. Thus, Kumar can no longer edit note since there is no write privilege for group and others. He can not get back the ownership either. But he can copy the file to his own directory, in which case he becomes the owner of the copy.

chgrp

This command changes the file’s group owner. No superuser permission is required. ls –l dept.lst

```
-rw-r--r--      1  kumar  metal  139  jun  8  16:43  dept.lst chgrp dba dept.lst; ls –l dept.lst
-rw-r--r--      1  kumar  dba  139  jun  8  16:43  dept.lst
```

In this chapter we considered two important file attributes – permissions and ownership. After we complete the first round of discussions related to files, we will take up the other file attributes.

Assignment

1 Mark Questions

1) What is a OS?
2) Why the name Linux has derived?
3) What is a notation used for current directory in Linux?
4) What is a notation used for Parent directory in Linux?
5) When was the first Linux has been released?
6) What is a significance of /bin directory?
7) What is the difference between cp and mv command?
8) How many levels of security has been found in Linux files system?
9) What is the use of *chmode* command?
10) What does *ls* command does?

7 Mark Questions

1) What is Linux? Give the characteristics of OS?
2) Describe the file system structure in Linux.
3) What is the difference between naming the file with absolute path and relative path? Illustrate with an example.
4) Give the different levels of security implementation available with Linux O.S.
5) What is the usage of *chown* and *chmod* commands in Linux?
6) List the different operations and their equivalent decimal values on a file for different users.
7) Give the features of Linux O.S.
8) Illustrate how to change different file permissions in Linux with example.
9) List out the functions of OS.
10) Illustrate how to change ownership of a file in Linux with example.

10 Mark Questions

1) In detail give the structure of Linux OS
2) Illustrate with examples how to change ownership as well as permissions of a file.