STEP-BY-STEP TUTORIAL FOR MASTERING LINUX | COMPLETE GUIDE FOR BECOMING A LINUX PROFESSIONAL

WRITTEN BY
CHRISTOPHER NEGUS
with contributions by
CHRISTINE BRESNAHAN

Linux® BIBLE

THE COMPREHENSIVE, TUTORIAL RESOURCE

BUILD LINUX DESKTOP AND SERVER SKILLS | ADVANCE TO ENTERPRISE-LEVEL COMPUTING | BECOME A LINUX SYSTEM ADMIN OR POWER USER

EIGHTH EDITION

Part I

Getting Started

IN THIS PART

Chapter 1
Starting with Linux

Chapter 2
Creating the Perfect Linux Desktop

CHAPTER

1

Starting with Linux

IN THIS CHAPTER

Learning what Linux is

Learning where Linux came from

Choosing Linux distributions

Exploring professional opportunities with Linux

Becoming certified in Linux

Linux is one of the most important technology advancements of the twenty-first century. Besides its impact on the growth of the Internet and its place as an enabling technology for a range of computer-driven devices, Linux development has been a model for how collaborative projects can surpass what single individuals and companies can do alone.

Google runs thousands upon thousands of Linux servers to power its search technology. Its Android phones are based on Linux. Likewise, when you download and run Google's Chrome OS, you get a browser that is backed by a Linux operating system.

Facebook builds and deploys its site using what is referred to as a LAMP stack (Linux, Apache web server, MySQL database, and PHP web scripting language) — all open source projects. In fact, Facebook itself uses an open source development model, making source code for the applications and tools that drive Facebook available to the public. This model has helped Facebook shake out bugs quickly, get contributions from around the world, and fuel Facebook's exponential growth.

Financial organizations that have trillions of dollars riding on the speed and security of their operating systems also rely heavily on Linux. These include the New York Stock Exchange, Chicago Mercantile Exchange, and the Tokyo Stock Exchange.

The widespread adoption of Linux around the world has created huge demand for Linux expertise. This chapter starts you on a path to becoming a Linux expert by helping you understand what Linux is, where it came from, and what your opportunities are for becoming proficient in it. The rest of this book provides you with hands-on activities to help you gain that expertise.

Understanding What Linux Is

Linux is a computer operating system. An operating system consists of the software that manages your computer and lets you run applications on it. The features that make up Linux and similar computer operating systems include the following:

- **Detecting and preparing hardware** — When the Linux system boots up (when you turn on your computer), it looks at the components on your computer (CPU, hard drive, network cards, and so on) and loads the software (drivers and modules) needed to access those particular hardware devices.
- **Managing processes** — The operating system must keep track of multiple processes running at the same time and decide which have access to the CPU and when. The system also must offer ways of starting, stopping, and changing the status of processes.
- **Managing memory** — RAM and swap space (extended memory) need to be allocated to applications as they need memory. The operating system decides how requests for memory are handled.
- **Providing user interfaces** — An operating system must provide ways of accessing the system. The first Linux systems were accessed from a command-line interpreter called the shell. Today, graphical desktop interfaces are commonly available as well.
- **Controlling filesystems** — Filesystem structures are built into the operating system (or loaded as modules). The operating system controls ownership and access to the files and directories that the filesystems contain.
- **Providing user access and authentication** — Creating user accounts and allowing boundaries to be set between users is a basic feature of Linux. Separate user and group accounts enable users to control their own files and processes.
- **Offering administrative utilities** — In Linux, there are hundreds (perhaps thousands) of commands and graphical windows to do such things as add users, manage disks, monitor the network, install software, and generally secure and manage your computer.
- **Starting up services** — To use printers, handle log messages, and provide a variety of system and network services, processes run in the background, waiting for requests to come in. There are many types of services that run in Linux. Linux provides different ways of starting and stopping these services. In other words, while Linux includes web browsers to view web pages, it can also be the computer that serves up web pages to others. Popular server features include web, mail, database, printer, file, DNS, and DHCP servers.
- **Programming tools** — A wide variety of programming utilities for creating applications and libraries for implementing specialty interfaces are available with Linux.

As someone managing Linux systems, you need to learn how to work with those features just described. While many features can be managed using graphical interfaces, an understanding of the shell command line is critical for someone administering Linux systems.

Modern Linux systems now go way beyond what the first UNIX systems (on which Linux was based) could do. Advanced features in Linux, often used in large enterprises, include the following:

- **Clustering** — Linux can be configured to work in clusters so that multiple systems can appear as one system to the outside world. Services can be configured to pass back and forth between cluster nodes, while appearing to those using the services that they are running without interruption.
- **Virtualization** — To manage computing resources more efficiently, Linux can run as a virtualization host. On that host, you could run other Linux systems, Microsoft Windows, BSD, or other operating systems as virtual guests. To the outside world, each of those virtual guests appears as a separate computer. KVM and Xen are two technologies in Linux for creating virtual hosts. Red Hat Enterprise Virtualization is a product from Red Hat, Inc. for managing multiple virtualization hosts, virtual guests, and storage.
- **Real-time computing** — Linux can be configured for real-time computing, where high-priority processes can expect fast, predictable attention.
- **Specialized storage** — Instead of just storing data on the computer's hard disk, a variety of specialized local and networked storage interfaces are available in Linux. Shared storage devices available in Linux include iSCSI, Fibre Channel, and Infiniband.

Many of these advanced topics are not covered in this book. However, the features covered here for using the shell, working with disks, starting and stopping services, and configuring a variety of servers should serve as a foundation for working with those advanced features.

Understanding How Linux Differs from Other Operating Systems

If you are new to Linux, chances are you have used a Microsoft Windows or Apple Mac OS operating system. Although Mac OS X has its roots in a free software operating system, referred to as the Berkeley Software Distribution (more on that later), operating systems from both Microsoft and Apple are considered proprietary operating systems. What that means is:

- You cannot see the code used to create the operating system.
- You, therefore, cannot change the operating system at its most basic levels if it doesn't suit your needs — and you can't use the operating system to build your own operating system from source code.

- You cannot check the code to find bugs, explore security vulnerabilities, or simply learn what that code is doing.
- You may not be able to easily plug your own software into the operating system if the creators of that system don't want to expose the programming interfaces you need to the outside world.

You might look at those statements about proprietary software and say, "What do I care? I'm not a software developer. I don't want to see or change how my operating system is built."

That may be true. But the fact that others can take free and open source software and use it as they please has driven the explosive growth of the Internet, mobile phones (think Android), special computing devices (think Tivo), and hundreds of technology companies. Free software has driven down computing costs and allowed for an explosion of innovation.

Maybe you don't want to use Linux — as Google, Facebook, and other companies have done — to build the foundation for a multi-billion dollar company. But those and other companies who now rely on Linux to drive their computer infrastructures are needing more and more people with the skills to run those systems.

You may wonder how a computer system that is so powerful and flexible has come to be free as well. To understand how that could be, you need to see where Linux came from. So, the next section of this chapter describes the strange and winding path of the free software movement that led to Linux.

Exploring Linux History

Some histories of Linux begin with this message posted by Linus Torvalds to the `comp.os.minix` newsgroup on August 25, 1991 (`http://groups.google.com/group/comp.os.minix/msg/b813d52cbc5a044b?pli=1`):

> Linus Benedict Torvalds
>
> Hello everybody out there using minix -
>
> I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons, among other things) . . . Any suggestions are welcome, but I won't promise I'll implement them :-)
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes — it's free of any minix code, and it has a multi-threaded fs. It is NOT protable[sic] (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Minix was a UNIX-like operating system that ran on PCs in the early 1990s. Like Minix, Linux was also a clone of the UNIX operating system. With few exceptions, such as Microsoft Windows, most modern computer systems (including Mac OS X and Linux) were derived from UNIX operating systems, created originally by AT&T.

To truly appreciate how a free operating system could have been modeled after a proprietary system from AT&T Bell Laboratories, it helps to understand the culture in which UNIX was created and the chain of events that made the essence of UNIX possible to reproduce freely.

NOTE

To learn more about how Linux was created, pick up the book *Just For Fun: The Story of an Accidental Revolutionary* by Linus Torvalds (Harper Collins Publishing, 2001).

Free-flowing UNIX culture at Bell Labs

From the very beginning, the UNIX operating system was created and nurtured in a communal environment. Its creation was not driven by market needs, but by a desire to overcome impediments to producing programs. AT&T, which owned the UNIX trademark originally, eventually made UNIX into a commercial product, but by that time, many of the concepts (and even much of the early code) that made UNIX special had fallen into the public domain.

If you are not old enough to remember when AT&T split up in 1984, you may not remember a time when AT&T was "the" phone company. Up until the early 1980s, AT&T didn't have to think much about competition because if you wanted a phone in the United States, you had to go to AT&T. It had the luxury of funding pure research projects. The mecca for such projects was the Bell Laboratories site in Murray Hill, New Jersey.

After a project called Multics failed in around 1969, Bell Labs employees Ken Thompson and Dennis Ritchie set off on their own to create an operating system that would offer an improved environment for developing software. Up to that time, most programs were written on punch cards that had to be fed in batches to mainframe computers. In a 1980 lecture on "The Evolution of the UNIX Time-sharing System," Dennis Ritchie summed up the spirit that started UNIX:

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

The simplicity and power of the UNIX design began breaking down barriers that, up until this point, had impeded software developers. The foundation of UNIX was set with several key elements:

- **The UNIX filesystem** — Because it included a structure that allowed levels of subdirectories (which, for today's desktop users, looks like folders inside of folders), UNIX could be used to organize the files and directories in intuitive ways.

Furthermore, complex methods of accessing disks, tapes, and other devices were greatly simplified by representing those devices as individual device files that you could also access as items in a directory.

- **Input/output redirection** — Early UNIX systems also included input redirection and pipes. From a command line, UNIX users could direct the output of a command to a file using a right-arrow key (>). Later, the concept of pipes (|) was added where the output of one command could be directed to the input of another command. For example, the following command line concatenates (`cat`) file1 and file2, sorts (`sort`) the lines in those files alphabetically, paginates the sorted text for printing (`pr`), and directs the output to the computer's default printer (`lpr`):

  ```
  $ cat file1 file2 | sort | pr | lpr
  ```

  This method of directing input and output enabled developers to create their own specialized utilities that could be joined with existing utilities. This modularity made it possible for lots of code to be developed by lots of different people. A user could just put together the pieces they needed.

- **Portability** — Simplifying the experience of using UNIX also led to it becoming extraordinarily portable to run on different computers. By having device drivers (represented by files in the filesystem tree), UNIX could present an interface to applications in such a way that the programs didn't have to know about the details of the underlying hardware. To later port UNIX to another system, developers had only to change the drivers. The application programs didn't have to change for different hardware!

To make portability a reality, however, a high-level programming language was needed to implement the software needed. To that end, Brian Kernighan and Dennis Ritchie created the C programming language. In 1973, UNIX was rewritten in C. Today, C is still the primary language used to create the UNIX (and Linux) operating system kernels.

As Ritchie went on to say in a 1979 lecture (`http://cm.bell-labs.com/who/dmr/hist.html`):

> Today, the only important UNIX program still written in assembler is the assembler itself; virtually all the utility programs are in C, and so are most of the application's programs, although there are sites with many in Fortran, Pascal, and Algol 68 as well. It seems certain that much of the success of UNIX follows from the readability, modifiability, and portability of its software that in turn follows from its expression in high-level languages.

If you are a Linux enthusiast and are interested in what features from the early days of Linux have survived, an interesting read is Dennis Ritchie's reprint of the first UNIX programmer's manual (dated November 3, 1971). You can find it at Dennis Ritchie's website: `http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html`. The form of this documentation is UNIX man pages — which is still the primary format for documenting UNIX and Linux operating system commands and programming tools today.

What's clear as you read through the early documentation and accounts of the UNIX system is that the development was a free-flowing process, lacked ego, and was dedicated to making UNIX excellent. This process led to a sharing of code (both inside and outside of Bell Labs), which allowed rapid development of a high-quality UNIX operating system. It also led to an operating system that AT&T would find difficult to reel back in later.

Commercialized UNIX

Before the AT&T divestiture in 1984, when it was split up into AT&T and seven "Baby Bell" companies, AT&T was forbidden to sell computer systems. Companies that would later become Verizon, Qwest, and Alcatel-Lucent were all part of AT&T. As a result of AT&T's monopoly of the telephone system, the U.S. government was concerned that an unrestricted AT&T might dominate the fledgling computer industry.

Because AT&T was restricted from selling computers directly to customers before its divestiture, UNIX source code was licensed to universities for a nominal fee. There was no UNIX operating system for sale from AT&T that you didn't have to compile yourself.

Berkeley Software Distribution arrives

In 1975, UNIX V6 became the first version of UNIX available for widespread use outside of Bell Laboratories. From this early UNIX source code, the first major variant of UNIX was created at University of California at Berkeley. It was named the Berkeley Software Distribution (BSD).

For most of the next decade, the BSD and Bell Labs versions of UNIX headed off in separate directions. BSD continued forward in the free-flowing, share-the-code manner that was the hallmark of the early Bell Labs UNIX, whereas AT&T started steering UNIX toward commercialization. With the formation of a separate UNIX Laboratory, which moved out of Murray Hill and down the road to Summit, New Jersey, AT&T began its attempts to commercialize UNIX. By 1984, divestiture was behind AT&T and it was ready to really start selling UNIX.

UNIX Laboratory and commercialization

The UNIX Laboratory was considered a jewel that couldn't quite find a home or a way to make a profit. As it moved between Bell Laboratories and other areas of AT&T, its name changed several times. It is probably best remembered by the name it had as it began its spin-off from AT&T: UNIX System Laboratories (USL).

The UNIX source code that came out of USL, the legacy of which is now owned in part by Santa Cruz Operation (SCO), has been used as the basis for ever-dwindling lawsuits by SCO against major Linux vendors (such as IBM and Red Hat, Inc.). Because of that, I think the efforts from USL that have contributed to the success of Linux are lost on most people.

During the 1980s, of course, many computer companies were afraid that a newly divested AT&T would pose more of a threat to controlling the computer industry than

would an upstart company in Redmond, Washington. To calm the fears of IBM, Intel, Digital Equipment Corporation, and other computer companies, the UNIX Lab made the following commitments to ensure a level playing field:

- **Source code only** — Instead of producing its own boxed set of UNIX, AT&T continued to sell only source code and to make it available equally to all licensees. Each company would then port UNIX to its own equipment. It wasn't until about 1992, when the lab was spun off as a joint venture with Novell (called Univel), and then eventually sold to Novell, that a commercial boxed set of UNIX (called UnixWare) was produced directly from that source code.
- **Published interfaces** — To create an environment of fairness and community to its OEMs (original equipment manufacturers), AT&T began standardizing what different ports of UNIX had to be able to do to still be called UNIX. To that end, Portable Operating System Interface (POSIX) standards and the AT&T UNIX System V Interface Definition (SVID) were specifications UNIX vendors could use to create compliant UNIX systems. Those same documents also served as road maps for the creation of Linux.

NOTE

In an early e-mail newsgroup post, Linus Torvalds made a request for a copy, preferably online, of the POSIX standard. I think that nobody from AT&T expected someone to actually be able to write his own clone of UNIX from those interfaces, without using any of its UNIX source code.

- **Technical approach** — Again, until the very end of USL, most decisions on the direction of UNIX were made based on technical considerations. Management was promoted up through the technical ranks, and to my knowledge, there was never any talk of writing software to break other companies' software or otherwise restrict the success of USL's partners.

When USL eventually started taking on marketing experts and creating a desktop UNIX product for end users, Microsoft Windows already had a firm grasp on the desktop market. Also, because the direction of UNIX had always been toward source-code licensing destined for large computing systems, USL had pricing difficulties for its products. For example, on software that it was including with UNIX, USL found itself having to pay out per-computer licensing fees that were based on $100,000 mainframes instead of $2,000 PCs. Add to that the fact that no application programs were available with UnixWare, and you can see why the endeavor failed.

Successful marketing of UNIX systems at the time, however, was happening with other computer companies. SCO had found a niche market, primarily selling PC versions of UNIX running dumb terminals in small offices. Sun Microsystems was selling lots of UNIX workstations (originally based on BSD but merged with UNIX in SVR4) for programmers and high-end technology applications (such as stock trading).

Other commercial UNIX systems were also emerging by the 1980s as well. This new ownership assertion of UNIX was beginning to take its toll on the spirit of open contributions. Lawsuits were being initiated to protect UNIX source code and trademarks. In 1984, this new, restrictive UNIX gave rise to an organization that eventually led a path to Linux: the Free Software Foundation.

GNU transitions UNIX to freedom

In 1984, Richard M. Stallman started the GNU project (`http://www.gnu.org`), recursively named by the phrase GNU is Not UNIX. As a project of the Free Software Foundation (FSF), GNU was intended to become a recoding of the entire UNIX operating system that could be freely distributed.

The GNU Project page (`http://www.gnu.org/gnu/thegnuproject.html`) tells the story of how the project came about in Stallman's own words. It also lays out the problems that proprietary software companies were imposing on those software developers who wanted to share, create, and innovate.

Although rewriting millions of lines of code might seem daunting for one or two people, spreading the effort across dozens or even hundreds of programmers made the project possible. Remember that UNIX was designed to be built in separate pieces that could be piped together. Because they were reproducing commands and utilities with well-known interfaces, that effort could easily be split among many developers.

It turned out that not only could the same results be gained by all new code, but in some cases, that code was better than the original UNIX versions. Because everyone could see the code being produced for the project, poorly written code could be corrected quickly or replaced over time.

If you are familiar with UNIX, try searching the thousands of GNU software packages for your favorite UNIX command from the Free Software Directory (`http://directory.fsf.org/GNU`). Chances are you will find it there, along with many, many other software projects available as add-ons.

Over time, the term *free software* has been mostly replaced by the term *open source software*. The term "free software" is preferred by the Free Software Foundation, while open source software is promoted by the Open Source Initiative (`http://www.opensource.org`).

To accommodate both camps, some people use the term *Free and Open Source Software* (FOSS) instead. An underlying principle of FOSS, however, is that, although you are free to use the software as you like, you have some responsibility to make the improvements you make to the code available to others. In that way, everyone in the community can benefit from your work as you have benefited from the work of others.

To clearly define how open source software should be handled, the GNU software project created the GNU Public License, or GPL. Although many other software licenses cover slightly different approaches to protecting free software, the GPL is the most well known — and it's the one that covers the Linux kernel itself. Basic features of the GNU Public License include the following:

- **Author rights** — The original author retains the rights to his or her software.
- **Free distribution** — People can use the GNU software in their own software, changing and redistributing it as they please. They do, however, have to include the source code with their distribution (or make it easily available).
- **Copyright maintained** — Even if you were to repackage and resell the software, the original GNU agreement must be maintained with the software, which means all future recipients of the software have the opportunity to change the source code, just as you did.

There is no warranty on GNU software. If something goes wrong, the original developer of the software has no obligation to fix the problem. However, many organizations, big and small, offer paid support packages for the software when it is included in their Linux or other open source software distribution. (See the "OSI open source definition" section later in this chapter for a more detailed definition of open source software.)

Despite its success in producing thousands of UNIX utilities, the GNU project itself failed to produce one critical piece of code: the kernel. Its attempts to build an open source kernel with the GNU Hurd project (`http://www.gnu.org/software/hurd`) were unsuccessful.

BSD loses some steam

The one software project that had a chance of beating out Linux to be the premier open source software project was the venerable old BSD project. By the late 1980s, BSD developers at University of California (UC) Berkeley realized that they had already rewritten most of the UNIX source code they had received a decade earlier.

In 1989, UC Berkeley distributed its own UNIX-like code as Net/1 and later (in 1991) as Net/2. Just as UC Berkeley was preparing a complete, UNIX-like operating system that was free from all AT&T code, AT&T hit them with a lawsuit in 1992. The suit claimed that the software was written using trade secrets taken from AT&T's UNIX system.

It's important to note here that BSD developers had completely rewritten the copyright-protected code from AT&T. Copyright was the primary means AT&T used to protect its rights to the UNIX code. Some believe that if AT&T had patented the concepts covered in that code, there might not be a Linux (or any UNIX clone) operating system today.

The lawsuit was dropped when Novell bought UNIX System Laboratories from AT&T in 1994. But, during that critical time period, there was enough fear and doubt about the legality of the BSD code that the momentum BSD had gained to that point in the fledgling open source community was lost. Many people started looking for another open

source alternative. The time was ripe for a college student from Finland who was working on his own kernel.

NOTE

Today, BSD versions are available from three major projects: FreeBSD, NetBSD, and OpenBSD. People generally characterize FreeBSD as the easiest to use, NetBSD as available on the most computer hardware platforms, and OpenBSD as fanatically secure. Many security-minded individuals still prefer BSD to Linux. Also, because of its licensing, BSD code can be used by proprietary software vendors, such as Microsoft and Apple, who don't want to share their operating system code with others. Mac OS X is built on a BSD derivative.

Linus builds the missing piece

Linus Torvalds started work on Linux in 1991, while he was a student at the University of Helsinki, Finland. He wanted to create a UNIX-like kernel so that he could use the same kind of operating system on his home PC that he used at school. At the time, Linus was using Minix, but he wanted to go beyond what the Minix standards permitted.

As noted earlier, Linus announced the first public version of the Linux kernel to the `comp.os.minix` newsgroup on August 25, 1991, although Torvalds guesses that the first version didn't actually come out until mid-September of that year.

Although Torvalds stated that Linux was written for the 386 processor and probably wasn't portable, others persisted in encouraging (and contributing to) a more portable approach in the early versions of Linux. By October 5, Linux 0.02 was released with much of the original assembly code rewritten in the C programming language, which made it possible to start porting it to other machines.

The Linux kernel was the last — and the most important — piece of code that was needed to complete a whole UNIX-like operating system under the GPL. So, when people started putting together distributions, the name Linux and not GNU is what stuck. Some distributions such as Debian, however, refer to themselves as GNU/Linux distributions. (Not including GNU in the title or subtitle of a Linux operating system is also a matter of much public grumbling by some members of the GNU project. See `http://www.gnu.org`.)

Today, Linux can be described as an open source UNIX-like operating system that reflects a combination of SVID, POSIX, and BSD compliance. Linux continues to aim toward compliance with POSIX as well as with standards set by the owner of the UNIX trademark, The Open Group (`http://www.unix.org`).

The non-profit Open Source Development Labs, renamed the Linux Foundation after merging with the Free Standards Group (`http://www.linuxfoundation.org`), which employs Linus Torvalds, manages the direction today of Linux development efforts. Its sponsors list is like a Who's Who of commercial Linux system and application vendors, including IBM, Red Hat, SUSE, Oracle, HP, Dell, Computer Associates, Intel, Cisco Systems, and others. The Linux Foundation's primary charter is to protect and accelerate

the growth of Linux by providing legal protection and software development standards for Linux developers.

Although much of the thrust of corporate Linux efforts is on corporate, enterprise computing, huge improvements are continuing in the desktop arena as well. The KDE and GNOME desktop environments continuously improve the Linux experience for casual users. Newer lightweight desktop environments such as Xfce and LXDE now offer efficient alternatives that today bring Linux to thousands of netbook owners.

Linus Torvalds continues to maintain and improve the Linux kernel.

NOTE

For a more detailed history of Linux, see the book *Open Sources: Voices from the Open Source Revolution* (O'Reilly, 1999). The entire first edition is available online at `http://oreilly.com/catalog/opensources/book/toc.html`.

OSI open source definition

Linux provides a platform that lets software developers change the operating system as they like and get a wide range of help creating the applications they need. One of the watchdogs of the open source movement is the Open Source Initiative (OSI, `http://www.opensource.org`).

Although the primary goal of open source software is to make source code available, other goals of open source software are also defined by OSI in its open source definition. Most of the following rules for acceptable open source licenses serve to protect the freedom and integrity of the open source code:

- **Free distribution** — An open source license can't require a fee from anyone who resells the software.
- **Source code** — The source code must be included with the software and there can be no restrictions on redistribution.
- **Derived works** — The license must allow modification and redistribution of the code under the same terms.
- **Integrity of the author's source code** — The license may require that those who use the source code remove the original project's name or version if they change the source code.
- **No discrimination against persons or groups** — The license must allow all people to be equally eligible to use the source code.
- **No discrimination against fields of endeavor** — The license can't restrict a project from using the source code because it is commercial or because it is associated with a field of endeavor that the software provider doesn't like.
- **Distribution of license** — No additional license should be needed to use and redistribute the software.

- **License must not be specific to a product** — The license can't restrict the source code to a particular software distribution.
- **License must not restrict other software** — The license can't prevent someone from including the open source software on the same medium as non–open source software.
- **License must be technology-neutral** — The license can't restrict methods in which the source code can be redistributed.

Open source licenses used by software development projects must meet these criteria to be accepted as open source software by OSI. More than 40 different licenses are accepted by OSI to be used to label software as "OSI Certified Open Source Software." In addition to the GPL, other popular OSI-approved licenses include:

- **LGPL** — The GNU Lesser General Public License (LGPL) is often used for distributing libraries that other application programs depend upon.
- **BSD** — The Berkeley Software Distribution License allows redistribution of source code, with the requirement that the source code keep the BSD copyright notice and not use the names of contributors to endorse or promote derived software without written permission. A major difference from GPL, however, is that BSD does not require people modifying the code to pass those changes on to the community. As a result, proprietary software vendors such as Apple and Microsoft have used BSD code in their own operating systems.
- **MIT** — The MIT license is like the BSD license, except that it doesn't include the endorsement and promotion requirement.
- **Mozilla** — The Mozilla license covers the use and redistribution of source code associated with the Firefox web browser and other software related to the Mozilla project (`http://www.mozilla.org`). It is a much longer license than the others just mentioned because it contains more definitions of how contributors and those reusing the source code should behave. This includes submitting a file of changes when submitting modifications and that those making their own additions to the code for redistribution should be aware of patent issues or other restrictions associated with their code.

The end result of open source code is software that has more flexibility to grow and fewer boundaries in how it can be used. Many believe that the fact that numerous people look over the source code for a project results in higher quality software for everyone. As open source advocate Eric S. Raymond says in an often-quoted line, "Many eyes make all bugs shallow."

Understanding How Linux Distributions Emerged

Having bundles of source code floating around the Internet that could be compiled and packaged into a Linux system worked well for geeks. More casual Linux users, however, needed a simpler way to put together a Linux system. To respond to that need, some of the best geeks began building their own Linux distributions.

A Linux distribution consists of the components needed to create a working Linux system and the procedures needed to get those components installed and running. Technically, Linux is really just what is referred to as the kernel. Before the kernel can be useful, you must have other software such as basic commands (GNU utilities), services you want to offer (such as remote login or web servers), and possibly a desktop interface and graphical applications. Then, you must be able to gather all that together and install it on your computer's hard disk.

Slackware (`http://www.slackware.org`) is one of the oldest Linux distributions still being developed today. It made Linux friendly for less technical users by distributing software already compiled and grouped into packages (those packages of software components were called *tarballs*). You would use basic Linux commands then to do things like format your disk, enable swap, and create user accounts.

Before long, many other Linux distributions were created. Some Linux distributions were created to meet special needs, such as KNOPPIX (a live CD Linux), Gentoo (a cool customizable Linux), and Mandrake (later called Mandriva, which was one of several desktop Linux distributions). But two major distributions rose to become the foundation for many other distributions: Red Hat Linux and Debian.

Choosing a Red Hat distribution

When Red Hat Linux appeared in the late 1990s, it quickly became the most popular Linux distribution for several reasons:

- **RPM package management** — While tarballs are fine for dropping software on your computer, they don't work as well when you want to update, remove, or even find out about that software. Red Hat created the RPM packaging format so a software package could contain not only the files to be shared, but also information about the package version, who created it, which files were documentation or configuration files, and when it was created. By installing software packaged in RPM format, that information about each software package could be stored in a local RPM database. It became easy to find what was installed, update it, or remove it.
- **Simple installation** — The anaconda installer made it much simpler to install Linux. As a user, you could step through some simple questions, in most cases accepting defaults, to install Red Hat Linux.
- **Graphical administration** — Red Hat added simple graphical tools to configure printers, add users, set time and date, and do other basic administrative tasks. As a result, desktop users could use a Linux system without even having to run commands.

For years, Red Hat Linux was the preferred Linux distribution for both Linux professionals and enthusiasts. Red Hat, Inc. gave away the source code, as well as the compiled,

ready-to-run versions of Red Hat Linux (referred to as the binaries). But as the needs of their Linux community users and big-ticket customers began to move further apart, Red Hat abandoned Red Hat Linux and began developing two operating systems instead: Red Hat Enterprise Linux and Fedora.

Using Red Hat Enterprise Linux

In March 2012, Red Hat, Inc. became the first open source software company to bring in more than $1 billion in yearly revenue. It achieved that goal by building a set of products around Red Hat Enterprise Linux (RHEL) that would suit the needs of the most demanding enterprise computing environments.

While other Linux distributions focused on desktop systems or small business computing, RHEL worked on those features needed to handle mission-critical applications for business and government. It built systems that could speed transactions for the world's largest financial exchanges and be deployed as clusters and virtual hosts.

Instead of just selling RHEL, Red Hat offers an ecosystem of benefits for Linux customers to draw on. To use RHEL, customers buy subscriptions that they can use to deploy any version of RHEL they desire. If they decommission a RHEL system, they can use the subscription to deploy another system.

Different levels of support are available for RHEL, depending on customer needs. Customers can be assured that, along with support, they can get hardware and third-party software that is certified to work with RHEL. They can get Red Hat consultants and engineers to help them put together the computing environments they need. They can also get training and certification exams for their employees (see the discussion of RHCE certification later in this chapter).

Red Hat has also added other products as natural extensions to Red Hat Enterprise Linux. JBoss is a middleware product for deploying Java-based applications to the Internet or company intranets. Red Hat Enterprise Virtualization is comprised of the virtualization hosts, managers, and guest computers that allow you to install, run, manage, migrate, and decommission huge virtual computing environments.

There are those who have tried to clone RHEL, using the freely available RHEL source code, rebuilding and rebranding it. CentOS is a community-sponsored Linux distribution that is built from RHEL source code. Oracle Linux is likewise built from source code for RHEL but currently offers an incompatible kernel. Despite that, however, RHEL is still the runaway leading enterprise computer operating system.

I've chosen to use Red Hat Enterprise Linux for many of the examples in this book because, if you want a career working on Linux systems, there is a huge demand for those who can administer RHEL systems. If you are starting out with Linux, however,

Fedora can provide an excellent, free entry point to the same skills you need to use and administer RHEL systems.

Using Fedora

While RHEL is the commercial, stable, supported Linux distribution, Fedora is the free, cutting-edge Linux distribution that is sponsored by Red Hat, Inc. Fedora is the Linux system Red Hat uses to engage the Linux development community and encourage those who want a free Linux for personal use.

Fedora includes more than 16,000 software packages, many of which keep up with the latest available open source technology. As a user, you can try the latest Linux desktop, server, and administrative interfaces in Fedora for free. As a software developer, you can create and test your applications using the latest Linux kernel and development tools.

Because the focus of Fedora is on the latest technology, it focuses less on stability. So expect that you might need to do some extra work to get everything working and that not all the software will be fully baked.

However, I recommend that you use Fedora for most of the examples in this book for the following reasons:

- Fedora is used as a proving ground for Red Hat Enterprise Linux. Red Hat tests many new applications in Fedora before committing them to RHEL. By using Fedora, you will learn the skills you need to work with features as they are being developed for Red Hat Enterprise Linux.
- For learning, Fedora is more convenient than RHEL, yet still includes many of the more advanced, enterprise-ready tools that are in RHEL.
- Fedora is free, not only as in "freedom" but also as in "you don't have to pay for it."

Fedora is extremely popular with those who develop open source software. However, in the past few years, another Linux distribution has captured the attention of many people starting out with Linux: Ubuntu.

Choosing Ubuntu or another Debian distribution

Like Red Hat Linux, the Debian GNU/Linux distribution was an early Linux distribution that excelled at packaging and managing software. Debian uses deb packaging format and tools to manage all of the software packages on its systems. Debian also has a reputation for stability.

Many Linux distributions can trace their roots back to Debian. According to distrowatch (`http://distrowatch.com`), more than 120 Linux distributions can be traced back to Debian. Popular Debian-based distributions include Linspire, Xandros, KNOPPIX, MEPIS, Damn Small Linux, and many others. However, the Debian derivative that has achieved the most success is Ubuntu (`http://www.ubuntu.com`).

By relying on stable Debian software development and packaging, the Ubuntu Linux distribution was able to come along and add those features that Debian lacked. In pursuit of bringing new users to Linux, the Ubuntu project added a simple graphical installer and easy-to-use graphical tools. It also focused on full-featured desktop systems, while still offering popular server packages.

Ubuntu was also an innovator in creating new ways to run Linux. Using live CDs offered by Ubuntu, you could have Ubuntu up and running in just a few minutes. Often included on those CDs were open source applications, such as web browsers and word processors, that actually ran in Windows. This made the transition to Linux from Windows easier for some people.

If you are using Ubuntu, don't fear. Most of subject matter covered in this book will work as well in Ubuntu as it does in Fedora or RHEL. As we enter some of the server sections of the book, however, you may find some of the enterprise-type, content may not match as exactly as it would on Fedora or RHEL.

Finding Professional Opportunities with Linux Today

If you want to develop an idea for a computer-related research project or technology company, where do you begin? You begin with an idea. After that, you look for the tools you need to explore and eventually create your vision. Then, you look for others to help you during that creation process.

Today, the hard costs of starting a company like Google or Facebook include just a computer, a connection to the Internet, and enough caffeinated beverage of your choice to keep you up all night writing code. If you have your own world-changing idea, Linux and thousands of software packages are available to help you build your dreams. The open source world also comes with communities of developers, administrators, and users who are available to help you.

If you want to get involved with an existing open source project, projects are always looking for people to write code, test software, or write documentation. In those projects, you will find people who use the software, work on the software, and are usually willing to share their expertise to help you as well.

But whether you seek to develop the next great open source software project or simply want to gain the skills needed to compete for the thousands of well-paying Linux administrator or development jobs, it will help you to know how to install, secure, and maintain Linux systems.

So, what are the prospects for Linux careers? A survey entitled "Linux Adoption Trends 2012: A Survey of Enterprise End Users" from the Linux Foundation (`http://www.linuxfoundation.org/publications/linux-foundation/linux-adoption-trends-end-user-report-2012`) surveyed more than

400 respondents from organizations with 500+ employees and more than $500 million in annual sales. Here is what the Linux Foundation found:

- **Increased Linux use** — More than 80 percent of the companies expect their use of Linux to increase over the next five years.
- **More Linux for big data** — More than 70 percent of the companies expect to add more Linux systems to handle big data (as compared to about 36 percent more Windows systems and 29 percent more UNIX systems). Big data refers to huge, complex, and unwieldy amounts of information that need to be stored and managed.
- **More Linux experts needed!** — Aside from some concerns for interoperability with existing platforms (37 percent), respondents' next biggest concern with Linux was being able to find talent to support those systems.

The major message to take from this survey is that Linux continues to grow and create demands for Linux expertise. Companies that have begun using Linux have continued to move forward with Linux. Those using Linux continue to expand its use and find that cost savings, security, and the flexibility it offers continue to make Linux a good investment.

Understanding how companies make money with Linux

Open source enthusiasts believe that better software can result from an open source software development model than from proprietary development models. So in theory, any company creating software for its own use can save money by adding its software contributions to those of others to gain a much better end product for themselves.

Companies that want to make money by selling software need to be more creative than they were in the old days. Although you can sell the software you create that includes GPL software, you must pass the source code of that software forward. Of course, others can then recompile that product, basically using and even reselling your product without charge. Here are a few ways that companies are dealing with that issue:

- **Software subscriptions** — Red Hat, Inc. sells its Red Hat Enterprise Linux products on a subscription basis. For a certain amount of money per year, you get binary code to run Linux (so you don't have to compile it yourself), guaranteed support, tools for tracking the hardware and software on your computer, access to the company's knowledge base, and other assets.

  Although Red Hat's Fedora project includes much of the same software and is also available in binary form, there are no guarantees associated with the software or future updates of that software. A small office or personal user might take a risk on using Fedora (which is itself an excellent operating system), but a big company that's running mission-critical applications will probably put down a few dollars for RHEL.
- **Training and certification** — With Linux system use growing in government and big business, professionals are needed to support those systems. Red Hat offers training courses and certification exams to help system administrators become proficient using Red Hat Enterprise Linux Systems. In particular, the

Red Hat Certified Engineer (RHCE) and Red Hat Certified System Administrator (RHCSA) certifications have become popular (`http://www.redhat.com/certification`). More on RHCE/RHCSA certifications later in this chapter.

Other certification programs are offered by Linux Professional Institute (`http://www.lpi.org`), CompTIA (`http://www.comptia.org`), and Novell (`http://www.novell.com`). LPI and CompTIA are professional computer industry associations. Novell centers its training and certification on its SUSE Linux products.

- **Bounties** — Software bounties are a fascinating way for open source software companies to make money. Let's say that you are using XYZ software package and you need a new feature right away. By paying a software bounty to the project itself, or to other software developers, you can have your needed improvements moved to the head of the queue. The software you pay for will remain covered by its open source license, but you will have the features you need, at probably a fraction of the cost of building the project from scratch.
- **Donations** — Many open source projects accept donations from individuals or open source companies that use code from their projects. Amazingly, many open source projects support one or two developers and run exclusively on donations.
- **Boxed sets, mugs, and T-shirts** — Many open source projects have online stores where you can buy boxed sets (some people still like physical CDs and hard copies of documentation) and a variety of mugs, T-shirts, mouse pads, and other items. If you really love a project, for goodness sake, buy a T-shirt!

This is in no way an exhaustive list, because more creative ways are being invented every day to support those who create open source software. Remember that many people have become contributors to and maintainers of open source software because they needed or wanted the software themselves. The contributions they make for free are worth the return they get from others who do the same.

Becoming Red Hat Certified

Although this book is not focused on becoming certified in Linux, it does touch on the activities you need to be able to master to pass popular Linux certification exams. In particular, most of what is covered in the Red Hat Certified Engineer (RHCE) and Red Hat Certified System Administrator (RHCSA) exams is described in this book.

If you are looking for a job as a Linux IT professional, often RHCSA or RHCE certification is listed as a requirement or at least a preference for employers. The RHCSA exam (EX200) provides the basic certification, covering such topics as configuring disks and filesystems, adding users, setting up a simple web and FTP server, and adding swap space. The RHCE exam (EX300) tests for more advanced server configuration, as well an advanced knowledge of security features, such as SELinux and firewalls (iptables).

Those of us who have taught RHCE/RHCSA courses and given exams (as I did for three years) are not allowed to tell you exactly what is on the exam. However, Red Hat does

give an overview of how the exams work, as well as a list of topics you could expect to see covered in the exam. You can find those exam objectives on the following sites:

- **RHCSA** — `http://www.redhat.com/training/courses/ex200/examobjective`
- **RHCE** — `http://www.redhat.com/training/courses/ex300/examobjective`

As the exam objectives state, the RHCSA and RHCE exams are performance-based, which means that you are given tasks to do and that you must perform those tasks on an actual Red Hat Enterprise Linux system, as you would on the job. You are graded on how well you obtained the results of those tasks.

If you plan to take the exams, check back to the exam objectives pages often, as they change from time to time. Keep in mind also that the RHCSA is a standalone certification; however, you must pass the RHCSA and the RHCE exams to get an RHCE certification. Often, the two exams are given on the same day.

You can sign up for RHCSA and RHCE training and exams at `http://training.redhat.com`. Training and exams are given at major cities all over the United States and around the world. The skills you need to complete these exams are described in the following sections.

RHCSA topics

As noted earlier, RHCSA exam topics cover basic system administration skills. These are the current topics listed at the RHCSA exam objectives site (again, check the exam objectives site in case they change):

- **Understand essential tools** — You are expected to have a working knowledge of the command shell (bash), including how to use proper command syntax and do input/output redirection (`< > >>`). You need to know how to log in to remote and local systems. Expect to have to create, move, copy, link, delete, and change permission and ownership on files. Likewise, you should know how to look up information on man pages and `/usr/share/doc`.
- **Operate running systems** — In this category, you must understand the Linux boot process, go into single-user mode, shut down, reboot, and change run levels. You must be able to start and stop virtual machines and network services, as well as find and interpret log files.
- **Configure local storage** — Setting up disk partitions includes creating physical volumes and configuring them to be used for Logical Volume Management (LVM) or encryption (LUKS). You should also be able to set up those partitions as filesystems or swap space that can be mounted or enabled at boot time.
- **Create and configure filesystems** — Create and automatically mount different kinds of filesystems, including regular Linux filesystems (ext2, ext3, or ext4), LUKS-encrypted filesystems, and network filesystems (NFS and CIFS). Create

collaborative directories using the set group ID bit feature and Access Control Lists (ACL). You must also be able to use LVM to extend the size of a logical volume.

- **Deploy, configure, and maintain systems** — This covers a range of topics, including configuring networking, creating `cron` tasks, setting the default run level, and installing RHEL systems. You must also be able to configure a simple HTTP and FTP server. For software packages, you must be able to install packages from Red Hat Network, a remote repository, or the local filesystem. Finally, you must be able to properly install a new kernel and choose that or some other kernel to boot up when the system starts.
- **Manage users and groups** — You must know how to add, delete, and change user and group accounts. Another topic you should know is password aging, using the `chage` command. You must also know how to configure a system to authenticate by connecting to an LDAP directory server.
- **Manage security** — You must have a basic understanding of how to set up a firewall (system-config-firewall or iptables) and how to use SELinux.

Most of these topics are covered in this book. Refer to Red Hat documentation (`http://docs.redhat.com`) under the Red Hat Enterprise Linux heading for descriptions of features not found in this book. In particular, the Deployment Guide contains descriptions of many of the RHCSA-related topics.

RHCE topics

RHCE exam topics cover more advanced server configuration, along with a variety of security features for securing those servers. Again, check the RHCE exam objectives site for the most up-to-date information on topics you should study for the exam.

The system configuration and management requirement for the RHCE exam covers a range of topics, including the following:

- **Route IP traffic** — Set up static routes to specific network addresses.
- **Firewalls** — Block or allow traffic to selected ports on your system that offer services such as web, FTP, and NFS, as well as block or allow access to services based on the originator's IP address.
- **Kernel tunables** — Set kernel tunable parameters using the `/etc/sysctl.conf` file and the `sysctl` command.
- **Configure iSCSI** — Set up system as an iSCSI initiator that mounts an iSCSI target at boot time.
- **System reports** — Use features such as sar to report on system use of memory, disk access, network traffic, and processor utilization.
- **Shell scripting** — Create a simple shell script to take input and produce output in various ways.

- **Remote logging** — Configure the `rsyslogd` facility to gather log messages and distribute them to a remote logging server. Also, configure a remote logging server facility to gather log messages from logging clients.
- **SELinux** — With Security Enhanced Linux in Enforcing mode, make sure that all server configurations described in the next section are properly secured with SELinux.

For each of the network services in the list that follows, make sure that you can go through the steps to install packages required by the service, set up SELinux to allow access to the service, set the service to start at boot time, secure the service by host or by user (using iptables, TCP wrappers, or features provided by the service itself), and configure it for basic operation. These are the services:

- **Web server** — Configure an Apache (HTTP/HTTPS) server. You must be able to set up a virtual host, deploy a CGI script, use private directories, and allow a particular Linux group to manage the content.
- **DNS server** — Set up a DNS server (bind package) to act as a caching-only name server that can forward DNS queries to another DNS server. No need to configure master or slave zones.
- **FTP server** — Set up an FTP server to provide anonymous downloads.
- **NFS server** — Configure an NFS server to share specific directories to specific client systems so they can be used for group collaboration.
- **Windows file sharing server** — Set up Linux (Samba) to provide SMB shares to specific hosts and users. Configure the shares for group collaboration.
- **Mail server** — Configure postfix or sendmail to accept incoming mail from outside of the local host. Relay mail to a smart host.
- **Secure Shell server** — Set up the SSH service (sshd) to allow remote login to your local system as well as key-based authentication. Otherwise, configure the `sshd.conf` file as needed.
- **Network Time server** — Configure a Network Time Protocol server (ntpd) to synchronize time with other NTP peers.

While there are other tasks in the RHCE exam, as just noted, keep in mind that most of the tasks have you configure servers, then secure those servers using any technique you need. Those can include firewall rules (iptables), SELinux, TCP Wrappers, or any features built into configuration files for the particular service.

Summary

Linux is an operating system that is built by a community of software developers around the world and led by its creator, Linus Torvalds. It is derived originally from the UNIX operating system, but has grown beyond UNIX in popularity and power over the years.

The history of the Linux operating system can be tracked from early UNIX systems that were distributed free to colleges and improved by initiatives such as the Berkeley Software Distribution (BSD). The Free Software Foundation helped make many of the components needed to create a fully-free UNIX-like operating system. The Linux kernel itself was the last major component needed to complete the job.

Most Linux software projects are protected by one of a set of licenses that fall under the Open Source Initiative umbrella. The most prominent of these is the GNU Public License (GPL). Standards such as the Linux Standard Base and world-class Linux organizations and companies (such as Red Hat, Inc.) make it possible for Linux to continue to be a stable, productive operating system into the future.

Learning the basics of how to use and administer a Linux system will serve you well in any aspect of working with Linux. The remaining chapters each provide a series of exercises with which you can test your understanding. That's why, for the rest of the book, you will learn best with a Linux system in front of you so you can work through the examples in each chapter and complete the exercises successfully.

The next chapter describes how to get started with Linux by describing how to get and use a Linux desktop system.

Contents

Introduction. xxxiii

Part I: Getting Started 1

Chapter 1: Starting with Linux 3

Understanding What Linux Is 4
Understanding How Linux Differs from Other Operating Systems 5
Exploring Linux History 6
Free-flowing UNIX culture at Bell Labs 7
Commercialized UNIX 9
Berkeley Software Distribution arrives 9
UNIX Laboratory and commercialization 9
GNU transitions UNIX to freedom 11
BSD loses some steam 12
Linus builds the missing piece 13
OSI open source definition 14
Understanding How Linux Distributions Emerged 15
Choosing a Red Hat distribution 16
Using Red Hat Enterprise Linux 17
Using Fedora 18
Choosing Ubuntu or another Debian distribution 18
Finding Professional Opportunities with Linux Today 19
Understanding how companies make money with Linux 20
Becoming Red Hat Certified 21
RHCSA topics 22
RHCE topics 23
Summary 24

Chapter 2: Creating the Perfect Linux Desktop. 27

Understanding Linux Desktop Technology 28
Starting with the Fedora GNOME Desktop Live CD 30
Using the GNOME 3 Desktop 31
After the computer boots up 31
Navigating with the mouse 32
Navigating with the keyboard 36
Setting up the GNOME 3 desktop 38

Extending the GNOME 3 desktop 39
Using GNOME shell extensions 39
Using the GNOME Tweak Tool 40
Starting with desktop applications 42
Managing files and folders with Nautilus 42
Installing and managing additional software 44
Playing music with Rhythmbox 45
Stopping the GNOME 3 desktop 46
Using the GNOME 2 Desktop 46
Using the Metacity window manager 48
Changing GNOME appearance 49
Using the GNOME panels 50
Using the Applications and System menus 51
Adding an applet 51
Adding another panel 52
Adding an application launcher 52
Adding a drawer 53
Changing panel properties 53
3D effects with AIGLX 54
Summary 57
Exercises 57

Part II: Becoming a Linux Power User 59

Chapter 3: Using the Shell 61

About Shells and Terminal Windows 62
Using the shell prompt 63
Using a terminal window 64
Using virtual consoles 65
Choosing Your Shell 65
Running Commands 66
Understanding command syntax 67
Locating commands 70
Recalling Commands Using Command History 72
Command-line editing 73
Command-line completion 75
Command-line recall 76
Connecting and Expanding Commands 78
Piping between commands 78
Sequential commands 79
Background commands 79
Expanding commands 80
Expanding arithmetic expressions 80

Expanding variables 81
Using Shell Variables 81
Creating and using aliases 83
Exiting the shell 84
Creating Your Shell Environment 84
Configuring your shell 84
Setting your prompt 85
Adding environment variables 87
Getting Information About Commands 88
Summary 90
Exercises 90

Chapter 4: Moving Around the Filesystem 93

Using Basic Filesystem Commands 96
Using Metacharacters and Operators 98
Using file-matching metacharacters 98
Using file-redirection metacharacters 99
Using brace expansion characters 101
Listing Files and Directories 101
Understanding File Permissions and Ownership 105
Changing permissions with chmod (numbers) 107
Changing permissions with chmod (letters) 107
Setting default file permission with umask 108
Changing file ownership 109
Moving, Copying, and Removing Files 110
Summary 111
Exercises 112

Chapter 5: Working with Text Files 113

Editing Files with vim and vi 113
Starting with vi 115
Adding text 115
Moving around in the text 116
Deleting, copying, and changing text 117
Pasting (putting) text 118
Repeating commands 118
Exiting vi 118
Skipping around in the file 119
Searching for text 120
Using ex mode 120
Learning more about vi and vim 120
Finding Files 121
Using locate to find files by name 121
Searching for files with find 122
Finding files by name 123

Finding files by size 124
Finding files by user 124
Finding files by permission 125
Finding files by date and time 126
Using not and or when finding files 126
Finding files and executing commands 127
Searching in files with grep 128
Summary 129
Exercises 130

Chapter 6: Managing Running Processes 131

Understanding Processes 131
Listing Processes 132
Listing processes with ps 132
Listing and changing processes with top 134
Listing processes with System Monitor 135
Managing Background and Foreground Processes 137
Starting background processes 138
Using foreground and background commands 139
Killing and Renicing Processes 140
Killing processes with kill and killall 140
Using kill to signal processes by PID 141
Using killall to signal processes by name 141
Setting processor priority with nice and renice 142
Summary 143
Exercises 143

Chapter 7: Writing Simple Shell Scripts 145

Understanding Shell Scripts 145
Executing and debugging shell scripts 146
Understanding shell variables 147
Special shell positional parameters 148
Reading in parameters 149
Parameter expansion in bash 149
Performing arithmetic in shell scripts 150
Using programming constructs in shell scripts 151
The “if . . . then” statements 151
The case command 154
The “for . . . do” loop 155
The “while . . . do” and “until . . . do” loops 156
Trying some useful text manipulation programs 157
The general regular expression parser 157
Remove sections of lines of text (cut) 158
Translate or delete characters (tr) 158

The stream editor (sed) 158
Using simple shell scripts 159
Telephone list 159
Backup script 160
Summary 161
Exercises 161

Part III: Becoming a Linux System Administrator 163

Chapter 8: Learning System Administration 165
Understanding System Administration 165
Using Graphical Administration Tools 167
Using the root User Account 169
Becoming root from the shell (su command) 170
Allowing administrative access via the GUI 171
Gaining administrative access with sudo 172
Exploring Administrative Commands, Configuration Files, and Log Files 174
Administrative commands 174
Administrative configuration files 175
Administrative log files 179
Using Other Administrative Accounts 180
Checking and Configuring Hardware 181
Checking your hardware 182
Managing removable hardware 184
Working with loadable modules 186
Listing loaded modules 187
Loading modules 187
Removing modules 188
Summary 188
Exercises 189

Chapter 9: Installing Linux 191
Choosing a Computer 192
Installing Fedora from a Live CD 193
Installing Red Hat Enterprise Linux from Installation Media 199
Installing Linux in the Enterprise 202
Exploring Common Installation Topics 204
Upgrading or installing from scratch 204
Dual booting 205
Installing Linux to run virtually 206
Using installation boot options 207
Boot options for disabling features 207
Boot options for video problems 208

Boot options for special installation types 208
Boot options for kickstarts and remote repositories 209
Miscellaneous boot options 210
Using specialized storage 210
Partitioning hard drives 211
Understanding different partition types 212
Partitioning during Fedora installation 212
Reasons for different partitioning schemes 216
Tips for creating partitions 216
Using the GRUB boot loader 218
Using GRUB Legacy (version 1) 218
Using GRUB 2 223
Summary 224
Exercises 225

Chapter 10: Getting and Managing Software 227

Managing Software with PackageKit 227
Enabling repositories and getting updates 228
Searching for packages 229
Installing and removing packages 230
Going beyond PackageKit 231
Understanding Linux RPM Software Packaging 231
Understanding RPM packaging 232
What is in an RPM? 233
Where do RPMs come from? 234
Installing RPMs 234
Managing RPM Packages with YUM 235
Understanding how yum works 235
1. Checking /etc/yum.conf 236
2. Checking /etc/sysconfig/rhn/up2date (RHEL only) 237
3. Checking /etc/yum.repos.d/*.repo files 237
4. Downloading RPM packages and metadata from a YUM repository 238
5. RPM packages installed to Linux file system 238
6. Store YUM repository metadata to local RPM database 238
Using YUM with third-party software repositories 239
Managing software with the YUM command 240
Searching for packages 240
Installing and removing packages 242
Updating packages 243
Updating groups of packages 244
Maintaining your RPM package database and cache 245
Downloading RPMs from a yum repository 246
Installing, Querying, and Verifying Software with the rpm Command 246
Installing and removing packages with rpm 247

Querying rpm information....247
Verifying RPM packages....249
Managing Software in the Enterprise....250
Summary....251
Exercises....252

Chapter 11: Managing User Accounts....253

Creating User Accounts....253
Adding users with useradd....256
Setting user defaults....259
Modifying users with usermod....260
Deleting users with userdel....261
Understanding Group Accounts....262
Using group accounts....262
Creating group accounts....263
Managing Users in the Enterprise....264
Setting permissions with Access Control Lists....265
Setting ACLs with setfacl....265
Setting default ACLs....267
Enabling ACLs....268
Adding directories for users to collaborate....270
Creating group collaboration directories (set GID bit)....270
Creating restricted deletion directories (sticky bit)....271
Centralizing User Accounts....272
Using the Authentication Configuration window....273
Summary....274
Exercises....275

Chapter 12: Managing Disks and Filesystems....277

Understanding Disk Storage....277
Partitioning Hard Disks....279
Viewing disk partitions....280
Creating a single-partition disk....281
Creating a multiple-partition disk....284
Using Logical Volume Management Partitions....288
Checking an existing LVM....288
Creating LVM logical volumes....291
Growing LVM logical volumes....293
Mounting Filesystems....293
Supported filesystems....294
Enabling swap areas....296
Disabling swap area....297
Using the fstab file to define mountable file systems....297
Using the mount command to mount file systems....300

Mounting a disk image in loopback....301
Using the umount command....301
Using the mkfs Command to Create a Filesystem....302
Summary....303
Exercises....303

Part IV: Becoming a Linux Server Administrator 305

Chapter 13: Understanding Server Administration....307

Starting with Server Administration....308
Step 1: Install the server....308
Step 2: Configure the server....310
Using configuration files....310
Checking the default configuration....310
Step 3: Start the server....311
Step 4: Secure the server....312
Password protection....313
Firewalls....313
TCP Wrappers....313
SELinux....313
Security settings in configuration files....314
Step 5: Monitor the server....314
Configure logging....314
Run system activity reports....314
Keep system software up to date....314
Check the filesystem for signs of crackers....315
Managing Remote Access with the Secure Shell Service....315
Starting the openssh-server service....316
Using SSH client tools....317
Using ssh for remote login....318
Using ssh for remote execution....319
Copying files between systems with scp and rsync....320
Interactive copying with sftp....323
Using key-based (passwordless) authentication....323
Configuring System Logging....325
Enabling system logging with rsyslog....325
Understanding the rsyslog.conf file....326
Understanding the messages log file....327
Setting up and using a loghost with rsyslogd....328
Watching logs with logwatch....329
Checking System Resources with sar....330
Checking System Space....332
Displaying system space with df....332

Checking disk usage with du 333
Finding disk consumption with find 333
Summary....334
Exercises 335

Chapter 14: Administering Networking....337

Configuring Networking for Desktops....338
Checking your network interfaces 340
Checking your network from NetworkManager.... 340
Checking your network from the command line.... 342
Configuring network interfaces.... 345
Configuring a network proxy connection 347
Configuring Networking for Servers 348
Using system-config-network 349
Choosing device configuration 350
Choosing DNS configuration 351
Understanding networking configuration files 351
Network interface files 352
Other networking files 353
Setting alias network interfaces 356
Setting up Ethernet channel bonding 357
Setting custom routes 358
Configuring Networking in the Enterprise 359
Configuring Linux as a router 359
Configuring Linux as a DHCP server 360
Configuring Linux as a DNS server 361
Configuring Linux as a proxy server 361
Configuring VLANs in Linux 362
Summary....363
Exercises 363

Chapter 15: Starting and Stopping Services 365

Understanding the Linux init Daemon 365
Understanding the classic init daemons....367
Understanding the Upstart init daemon....375
Learning Upstart init daemon basics 375
Learning Upstart's backward compatibility to SysVinit....378
Understanding systemd init 382
Learning systemd basics 382
Learning systemd's backward compatibility to SysVinit....388
Auditing Services....390
Auditing the classic SysVinit daemon 391
Auditing the Upstart init daemon....392
Auditing the systemd init....393

Stopping and Starting Services 394
Stopping and starting the classic SysVinit daemon....395
Stopping and starting the Upstart init daemon 396
Stopping and starting the systemd daemon 397
Stopping a service with systemd....397
Starting a service with systemd....398
Restarting a service with systemd 398
Reloading a service with systemd....399
Configuring Persistent Services 400
Configuring the classic SysVinit daemon persistent services 400
Configuring Upstart init daemon persistent services....401
Configuring systemd init persistent services.... 402
Enabling a service with systemd 402
Disabling (removing) a service with systemd.... 402
Configuring a Default runlevel or target unit 404
Configuring the classic SysVinit daemon default runlevel 404
Configuring the Upstart init daemon default runlevel 404
Configuring the systemd init default target unit 405
Adding New or Customized Services 406
Adding new services to classic SysVinit daemon.... 406
Step 1: Create a new or customized service script file.... 406
Step 2: Move the service script 407
Step 3: Add the service to runlevels 407
Adding new services to the Upstart init daemon.... 408
Adding new services to systemd init 410
Step 1: Create a new or customized service configuration unit file 410
Step 2: Move the service configuration unit file....411
Step 3: Add the service to the Wants directory 412
Summary....413
Exercises 413

Chapter 16: Configuring a Print Server. 415

Common UNIX Printing System 415
Setting Up Printers....417
Adding a printer automatically 417
Using web-based CUPS administration....418
Using the Printer Configuration window 420
Configuring local printers with the Printer Configuration window 421
Configuring remote printers 424
Adding a remote CUPS printer.... 425
Adding a remote UNIX (LDP/LPR) printer.... 425
Adding a Windows (SMB) printer 426
Working with CUPS Printing....427
Configuring the CUPS server (cupsd.conf) 427
Starting the CUPS server 429

Configuring CUPS printer options manually 429
Using Printing Commands 431
Printing with lpr 431
Listing status with lpc 431
Removing print jobs with lprm 432
Configuring Print Servers 433
Configuring a shared CUPS printer 433
Configuring a shared Samba printer 435
Understanding smb.conf for printing 435
Setting up SMB clients 436
Summary 437
Exercises 437

Chapter 17: Configuring a Web Server 439

Understanding the Apache Web Server 439
Getting and Installing Your Web Server 440
Understanding the httpd package 440
Installing Apache 443
Starting Apache 443
Securing Apache 444
Apache file permissions and ownership 445
Apache and iptables 445
Apache and SELinux 445
Understanding the Apache configuration files 446
Using directives 447
Understanding default settings 449
Adding a virtual host to Apache 451
Allowing users to publish their own web content 453
Securing your web traffic with SSL/TLS 455
Understanding how SSL is configured 456
Generating an SSL key and self-signed certificate 458
Generating a certificate signing request 459
Troubleshooting Your Web Server 460
Checking for configuration errors 460
Accessing forbidden and server internal errors 463
Summary 464
Exercises 464

Chapter 18: Configuring an FTP Server 467

Understanding FTP 467
Installing the vsftpd FTP Server 469
Starting the vsftpd Service 470
Securing Your FTP Server 472
Opening up your firewall for FTP 473
Allowing FTP access in TCP wrappers 474

Configuring SELinux for your FTP server 475
Relating Linux file permissions to vsftpd 476
Configuring Your FTP Server 477
Setting up user access 477
Allowing uploading 478
Setting up vsftpd for the Internet 479
Using FTP Clients to Connect to Your Server 481
Accessing an FTP server from Firefox 481
Accessing an FTP server with the lftp command 482
Using the gFTP client 484
Summary 485
Exercises 485

Chapter 19: Configuring a Windows File Sharing (Samba) Server 487

Understanding Samba 487
Installing Samba 488
Starting and Stopping Samba 490
Starting the Samba (smb) service 490
Starting the NetBIOS (nmbd) name server 492
Stopping the Samba (smb) and NetBIOS (nmb) services 493
Securing Samba 494
Configuring firewalls for Samba 495
Configuring SELinux for Samba 496
Setting SELinux Booleans for Samba 496
Setting SELinux file contexts for Samba 497
Configuring Samba host/user permissions 498
Configuring Samba 498
Using system-config-samba 498
Choosing Samba server settings 499
Configuring Samba user accounts 500
Creating a Samba shared folder 501
Checking the Samba share 502
Configuring Samba in the smb.conf file 503
Configuring the [global] section 504
Configuring the [homes] section 505
Configuring the [printers] section 506
Creating custom shared directories 507
Accessing Samba Shares 509
Accessing Samba shares in Linux 509
Accessing Samba shares in Windows 512
Using Samba in the Enterprise 512
Summary 513
Exercises 513

Chapter 20: Configuring an NFS File Server 515

Installing an NFS Server 517
Starting the NFS service 518
Sharing NFS Filesystems 519
Configuring the /etc/exports file 520
Hostnames in /etc/exports 521
Access options in /etc/exports 522
User mapping options in /etc/exports 522
Exporting the shared filesystems 523
Securing Your NFS Server 523
Opening up your firewall for NFS 524
Allowing NFS access in TCP wrappers 525
Configuring SELinux for your NFS server 526
Using NFS Filesystems 527
Viewing NFS shares 527
Manually mounting an NFS filesystem 527
Mounting an NFS filesystem at boot time 528
Mounting noauto filesystems 529
Using mount options 530
Using autofs to mount NFS filesystems on demand 532
Automounting to the /net directory 532
Automounting home directories 533
Unmounting NFS filesystems 535
Summary 536
Exercises 536

Chapter 21: Troubleshooting Linux 539

Boot-Up Troubleshooting 539
Starting from the BIOS 540
Troubleshooting BIOS setup 541
Troubleshooting boot order 542
Troubleshooting the GRUB boot loader 542
Starting the kernel 545
Troubleshooting the init process 546
Troubleshooting rc.sysinit 546
Troubleshooting runlevel processes 547
Troubleshooting Software Packages 551
Fixing RPM databases and cache 555
Troubleshooting Networking 556
Troubleshooting outgoing connections 556
View network interfaces 557
Check physical connections 557
Check routes 557

Check hostname resolution 558
Troubleshooting incoming connections 560
Check if the client can reach your system at all.... 560
Check if the service is available to the client 560
Check the firewall on the server 561
Check the service on the server 562
Troubleshooting Memory 563
Uncovering memory issues 563
Checking for memory problems 566
Dealing with memory problems 567
Troubleshooting in Rescue Mode 568
Summary....569
Exercises 570

Part V: Learning Linux Security Techniques 571

Chapter 22: Understanding Basic Linux Security 573

Introducing the Security Process Lifecycle....573
Examining the Planning Phase....575
Choosing an access control model....575
Discretionary Access Control....575
Mandatory Access Control....576
Role Based Access Control....576
Using security checklists 577
Access Control Matrix....577
Industry security checklists 578
Entering the Implementation Phase 578
Implementing physical security 578
Implementing disaster recovery 579
Securing user accounts.... 580
One user per user account.... 580
No logins to the root account 581
Setting expiration dates on temporary accounts 582
Removing unused user accounts 583
Securing passwords....585
Choosing good passwords 585
Setting and changing passwords 586
Enforcing best password practices 587
Understanding the password files and password hashes....590
Securing the filesystem....591
Managing dangerous filesystem permissions 591
Securing the password files....592
Locking down the filesystem....594

Managing software and services 595
Removing unused software and services 595
Updating software packages 596
Advanced implementation 596
Working in the Monitoring Phase 596
Monitoring log files 596
Monitoring user accounts 600
Detecting counterfeit new accounts and privileges 600
Detecting bad account passwords 602
Monitoring the filesystem 603
Verifying software packages 604
Scanning the filesystem 605
Detecting viruses and rootkits 606
Detecting an intrusion 608
Working in the Audit/Review Phase 611
Conducting compliance reviews 611
Conducting security reviews 612
Summary 612
Exercises 613

Chapter 23: Understanding Advanced Linux Security 615

Implementing Linux Security with Cryptography 615
Understanding hashing 616
Understanding encryption/decryption 618
Understanding cryptographic ciphers 618
Understanding cryptographic cipher keys 619
Understanding digital signatures 625
Implementing Linux cryptography 627
Ensuring file integrity 627
Encrypting a Linux filesystem 628
Encrypting a Linux directory 630
Encrypting a Linux file 633
Encrypting Linux miscellaneous 634
Implementing Linux Security with PAM 635
Understanding the PAM authentication process 636
Understanding PAM contexts 638
Understanding PAM control flags 638
Understanding PAM modules 639
Understanding PAM system event configuration files 640
Administering PAM on your Linux system 641
Managing PAM-aware application configuration files 641
Managing PAM system event configuration files 642
Implementing resources limits with PAM 644
Implementing time restrictions with PAM 646

Enforcing good passwords with PAM 648
Encouraging sudo use with PAM 652
Locking accounts with PAM 653
Obtaining more information on PAM 655
Summary 656
Exercises 656

Chapter 24: Enhancing Linux Security with SELinux 659

Understanding SELinux Benefits 659
Understanding How SELinux Works 661
Understanding Type Enforcement 661
Understanding Multi-Level Security 662
Implementing SELinux security models 663
Understanding SELinux Operational Modes 663
Understanding SELinux security contexts 664
Understanding SELinux Policy types 667
Understanding SELinux Policy rule packages 668
Configuring SELinux 669
Setting the SELinux Operational Mode 670
Setting the SELinux Policy type 672
Managing SELinux security contexts 673
Managing the user security context 674
Managing the file security context 675
Managing the process security context 676
Managing SELinux policy rule packages 676
Managing SELinux via Booleans 678
Monitoring and Troubleshooting SELinux 679
Understanding SELinux logging 679
Reviewing SELinux messages in the audit log 680
Reviewing SELinux messages in the messages log 680
Troubleshooting SELinux logging 682
Troubleshooting common SELinux problems 682
Using a non-standard directory for a service 683
Using a non-standard port for a service 683
Moving files and losing security context labels 684
Booleans set incorrectly 684
Putting It All Together 684
Obtaining More Information on SELinux 685
Summary 686
Exercises 686

Chapter 25: Securing Linux on a Network 689

Auditing Network Services 690
Evaluating access to network services 692
Using nmap to create a network services list 692

Using nmap to audit your network services advertisements....695
Controlling access to network services....699
Working with Firewalls....702
Understanding firewalls....702
Implementing firewalls....703
Understanding the iptables utility....703
Using the iptables utility....707
Summary....715
Exercises....716

Part VI: Appendixes 717

Appendix A: Media....719
Getting Fedora....720
Getting Red Hat Enterprise Linux....721
Getting Ubuntu....722
Creating Linux CDs and DVDs....724
Burning CDs/DVDs in Windows....724
Burning CDs/DVDs on a Mac OS X system....724
Burning CDs/DVDs in Linux....725
Burning CDs from a Linux desktop....725
Burning CDs from a Linux command line....726
Booting Linux from a USB Drive....727

Appendix B: Exercise Answers....729
Chapter 2: Creating the Perfect Linux Desktop....729
Chapter 3: Using the Shell....732
Chapter 4: Moving Around the Filesystem....734
Chapter 5: Working with Text Files....735
Chapter 6: Managing Running Processes....737
Chapter 7: Writing Simple Shell Scripts....738
Chapter 8: Learning System Administration....740
Chapter 9: Installing Linux....743
Chapter 10: Getting and Managing Software....745
Chapter 11: Managing User Accounts....746
Chapter 12: Managing Disks and Filesystems....750
Chapter 13: Understanding Server Administration....752
Chapter 14: Administering Networking....755
Chapter 15: Starting and Stopping Services....758
Chapter 16: Configuring a Print Server....761
Chapter 17: Configuring a Web Server....763
Chapter 18: Configuring an FTP Server....766
Chapter 19: Configuring a Windows File Sharing (Samba) Server....769
Chapter 20: Configuring an NFS File Server....772

Chapter 21: Troubleshooting Linux 774
Chapter 22: Understanding Basic Linux Security 776
Chapter 23: Understanding Advanced Linux Security 777
Chapter 24: Enhancing Linux Security with SELinux 779
Chapter 25: Securing Linux on a Network 781

Index 783

Your definitive guide to becoming a Linux expert

As a bestselling Linux author and full-time trainer for Red Hat, Christopher Negus has helped thousands of beginning and experienced Linux users become certified professionals. In this full updated edition of the popular *Linux Bible,* Negus and contributing author Christine Bresnahan give you a thorough Linux tutorial, complete with helpful exercises at the end of each chapter.

This new *Linux Bible* is a great hands-on tool and reference that will take you from beginner to power user.

If you want to...

- Learn Linux, but have never used it before
- Acquire the foundation to become a certified Linux professional
- Start on a career path that will last for decades
- Master skills you can use with all Linux distributions

...this is the book for you.

Learn how to:

- Install, set up, and use powerful Linux systems for desktops and servers
- Configure the perfect Linux desktop system
- Perform critical system administration tasks
- Set up your own print, file, and Web servers
- Get a stable and secure system using Linux security tools
- Move into enterprise-level computing

Chris Negus is an instructor for Red Hat, Inc. and the author of dozens of Linux and UNIX books, including *Red Hat Linux Bible* (all editions), *CentOS Bible*, *Fedora Bible*, *Ubuntu Linux Toolbox*, *Linux Troubleshooting Bible*, *Linux Toys*, and *Linux Toys II*. **Christine Bresnahan** has over 25 years' experience as a system administrator. She is an adjunct professor at Ivy Tech Community College, teaching Linux system administration, Linux security, and Windows security classes. She co-authored the *Linux Command Line and Shell Scripting Bible, 2nd Edition*.

Start with any Linux system and advance to enterprise Linux computing

- Use your favorite Linux distribution to learn and test your skills with Linux command-line tools
- Learn professional system administration tasks using Fedora, Red Hat Enterprise Linux, or other enterprise-ready Linux systems

Cover image: © Aleksandar Velasevic / iStockPhoto

ISBN 978-1-118-21854-9

54999

9 781118 218549

SHELVING CATEGORY:
Computers / Operating Systems / Linux

READER LEVEL:
Beginning to Advanced

WILEY
wiley.com

Also available as an e-book