Seventh Edition

- Set up, maintain, and secure the latest distributions of Linux
- Configure and manage file systems, as well as e-mail, FTP, DNS, web, VoIP, LDAP, and print servers
- Build and deploy virtual machines, containers, and servers

Linux Administration

A Beginner's Guide

Wale Soyinka

Mc Graw Hill Education

Linux Administration

A Beginner's Guide
Seventh Edition

WALE **SOYINKA**

Mc Graw Hill Education

New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

ISBN: 978-0-07-184537-3

MHID: 0-07-184537-2

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-184536-6,
MHID: 0-07-184536-4.

eBook conversion by codeMantra
Version 1.0

Dedicated to everyone who has contributed to open source technologies and ideals in one form or another. Without you, we would have nothing to write about in this book.

About the Author

Wale Soyinka is a systems/network engineering consultant and has written a decent library of Linux administration training materials. In addition to several editions of *Linux Administration: A Beginner's Guide*, he is the author of *Wireless Network Administration: A Beginner's Guide* and a projects lab manual, *Microsoft Windows 2000 Managing Network Environments* (Prentice Hall). Wale participates in several open source discussions, projects, and ventures—usually all centered around promoting and showcasing open source technologies and culture.

About the Contributor

Buki Adeniji is a seasoned telephony and telecomms professional who is extremely passionate about Linux, VoIP, and Asterisk, and has many global deployments under his belt. His core skill set was developed by working on key telephone systems (KTS), after which he mastered TDM/IP PBXs, including the Ericsson MD110, the most advanced TDM PBX at that time. Buki also has experience building and managing high-availability carrier-grade VoIP platforms. He designed interactive voice response (IVR) and other voice-based solutions while heading global operations teams that managed mission-critical cloud-based services, achieving 99.99 percent ("four nines") service availability. Being a lifetime learner with advanced degrees in Telecomms, Business, and Project Management, Buki's goal is to continue to excel at the boundaries of technology and innovation by using principles of ethnography and design thinking to develop technical solutions that address current and next-generation business needs. Above all, Buki is passionate about converting complex concepts and procedures into simple, concise, and executable steps. Buki currently resides in the tech hub of the Bay Area, where he is a passionate family man, as well as a maker who loves to explore the gadget world and the outdoors.

About the Technical Editor

David Lane is an infrastructure architect and IT manager working and living in the Washington, DC, area. He has been working with open source software since the early 1990s and was introduced to Linux via the Slackware distribution early in its development. David soon discovered Red Hat 3 and has never looked back. Unlike most Linux people, David does not have a programming background and fell into IT as a career after discovering he was not cut out for sleeping on the street. He has implemented Linux solutions for a variety of government and private companies with solutions ranging from the simple to the complex. In his spare time, David writes about open source issues, especially those related to business, for the *Linux Journal*, as well as championing Linux to the next generation. David is an amateur radio operator and Emergency Coordinator for amateur radio responders for his local county. David speaks regularly to both Linux and amateur radio user groups about the synergies between open source and amateur radio.

At a Glance

Contents

PART III
Networking and Security

PART VI

Appendixes

Acknowledgments

My acknowledgment list is a very long and philosophical one. It includes everybody who has ever believed in me and provided me with one opportunity or another to experience various aspects of my life up to this point. It includes everybody I have ever had any kind of direct or indirect contact with. It includes everyone I have ever had a conversation with. It includes everybody I have ever looked at. It includes everyone who has ever given to or taken away from me. You have all contributed to and enriched my life. I am me because of you. You know who you are, and I thank you.

This page intentionally left blank

Introduction

On October 5, 1991, Linus Torvalds posted this message to the news-group comp.os.minix:

```
Do you pine for the nice days of minix-1.1, when men were men
and wrote their own device drivers? Are you without a nice
project and just dying to cut your teeth on an OS you can try
to modify for your needs? Are you finding it frustrating when
everything works on minix? No more all-nighters to get a nifty
program working? Then this post might be just for you :-)
```

Linus went on to introduce the first cut of Linux to the world. Unbeknown to him, he had unleashed what was to become one of the world's most popular and disruptive operating systems. More than 24 years later, an entire industry has grown up around Linux. And, chances are, you've probably already used it (or benefitted from it) in one form or another!

Who Should Read This Book

A part of the title of this book reads "A Beginner's Guide"; this is mostly apt. But what the title should say is "A Beginner to Linux Administration Guide," because we do make a few assumptions about you, the reader. (And we jolly well couldn't use that title because it was such a mouthful and not sexy enough.) We assume that you are already familiar with Microsoft Windows servers (or other operating systems, but we'll assume Windows for the sake of discussion) at a "power user" level or better. We assume that you are familiar with the terms (and some concepts) necessary to run a small- to medium-sized network of computers. Any experience with bigger networks or advanced Windows technologies, such as Active Directory, will allow you to get more from the book but is not required.

We make these assumptions because we did not want to write a guide for dummies. There are already enough books on the market that tell you what to click without telling you why; this book is not meant to be among those ranks.

In addition to your Windows background, we assume that you're interested in having more information about the topics here than the material we have written alone. After all, we've spent only 30 to 35 pages on topics that have entire books devoted to them! For this reason, we have scattered references to other resources throughout the chapters. We urge you to take advantage of these recommendations.

We believe that seasoned Linux system administrators can also benefit from this book because it can serve as a quick how-to cookbook on various topics that might not be the seasoned reader's strong points. We understand that system administrators generally have aspects of system administration that they like or loath a lot. For example, making backups is not one of our favorite aspects of system administration, and this is reflected in the half a page we've dedicated to backups. (Just kidding, there's an entire chapter on the topic.)

What's in This Book?

Linux Administration: A Beginner's Guide, Seventh Edition comprises six parts.

Part I: Introduction, Installation, and Software Management

Part I includes four chapters (Chapter 1, "Technical Summary of Linux Distributions"; Chapter 2, "Installing Linux in a Server Configuration"; Chapter 3, "The Command Line" ; and Chapter 4, "Managing Software"). The first two chapters provide you with a nice overview of what Linux is, how it compares to Windows in several key areas, and how to install server-grade Fedora and Ubuntu Linux distributions.

Chapter 3, "The Command Line," begins covering the basics of working with the Linux command-line interface (CLI) so that you can become comfortable working without a GUI. Although it is possible to administer a system from within the graphical desktop, your greatest power comes from being comfortable with both the CLI and the GUI. (This is true for Windows, too. Don't believe that? Open a command prompt, run **netsh**, and try to do what **netsh** does in the GUI.)

Part I ends with a chapter on how to install software from prepackaged binaries and source code, as well as how to perform standard software management tasks.

Ideally, the information in Part I should be enough information to get you started and help you draw parallels to how Linux works based on your existing knowledge of other operating systems. Some of the server installation and software installation tasks performed in Part I help serve as a reference point for some other parts of the book.

Part II: Single-Host Administration

Part II covers the material necessary to manage what we call a "stand-alone" system, a system that does not require or provide any services to other systems on the network. Although the notion of a server *not* serving anything might seem counterintuitive at first, it is the foundation on which many other concepts are built, and it will come in handy for your understanding of network-based services later on.

Part II comprises six chapters. Chapter 5, "Managing Users and Groups," covers the underlying basics of user and group concepts on Linux platforms, as well as day-to-day management tasks of adding and removing users and groups. The chapter also introduces the basic concepts of multiuser operation and the Linux permissions model.

Chapter 6, "Booting and Shutting Down," documents the entire booting and shutting down processes. This chapter includes details on how to start up services properly and shut them down properly. You'll learn how to add new services manually, which will also come in handy later on in the book.

Chapter 7, "File Systems," continues with the basics of file systems—their organization, creation, and, most important, their management.

The basics of operation continue in Chapter 8, "Core System Services," with coverage of basic tools such as **`xinetd`**, **`upstart`**, **`rsyslog`**, **`cron`**, **`systemd`**, **`journald`**, and so on. **`xinetd`** is the Linux equivalent of Windows' **`svchost`**, and **`rsyslog`** manages logging for all applications in a unified framework. You might think of **`rsyslog`** and **`journald`** as more flexible versions of the Windows Event Viewer.

Kernel coverage in Chapter 9, "The Linux Kernel," documents the process of configuring, compiling, and installing your own custom kernel in Linux. This capability is one of the points that gives Linux administrators an extraordinary amount of fine-grained control over how their systems operate.

Chapter 10, "Knobs and Dials: Virtual File Systems," covers some kernel-level tweaking through the **/proc** and **/sys** file systems. The ability to view and modify certain kernel-level configuration and runtime variables through **/proc** and **/sys**, as shown in this chapter, gives administrators almost infinite kernel fine-tuning possibilities. When applied properly, this ability amounts to an arguably better and easier way to tweak the kernel than is afforded by Windows platforms.

Part III: Networking and Security

Part III begins our journey into the world of networking and security. We are not quite sure why we combined these two things together under the same part, but if forced to justify this, we would say that "the network is the root of most evils and therefore it needs to be secured." Moving right along; with the ongoing importance of security on the Internet, as well as compliance issues with Sarbanes-Oxley (SOX) and the Health Insurance Portability and Accountability Act (HIPAA), the use of Linux in scenarios that require high security is unprecedented.

We deliberately decided to cover security before introducing network-based services (Parts IV and V) so that we could touch on some essential security best practices that can help in protecting your network-based services from attacks.

This section kicks off with Chapter 11, "TCP/IP for System Administrators," which provides a detailed overview of TCP/IP in the context of what system (and/or network) administrators need to know. The chapter provides a lot of detail on how to use troubleshooting tools such as **`tcpdump`** to capture packets and read them back, as well as a step-by-step analysis of how TCP connections work. These tools should enable you to troubleshoot network peculiarities effectively.

Chapter 12, "Network Configuration," returns to administration issues by focusing on basic network configuration (for both IPv4 and IPv6). This includes setting up IP addresses, routing entries, and so on. We extend past the basics in Chapter 13, "Linux Firewall (Netfilter)," by delving into advanced networking concepts and showing you how to build a Linux-based firewall and router. We touch on the new **nftables** project, which is slated to replace the existing and popular **iptables** framework.

Chapter 14, "Local Security," and Chapter 15, "Network Security," discuss aspects of system and network security in detail. They include Linux-specific issues as well as general security tips and tricks and commonsensical stuff so that you can better configure your system and protect it against attacks.

Part IV: Internet Services

The remainder of the book is divided into two distinct parts: "Internet Services" and "Intranet Services." Although they sound similar, they are different—InTER(net) and InTRA(net). We define Internet services as those running on a Linux system exposed directly to the Internet. Examples include web and Domain Name System (DNS) services. We define intranet services as those that are typically run behind a firewall for internal users and mostly for internal consumption only.

This section starts off with Chapter 16, "Domain Name System (DNS)," which covers the information you need to know to install, configure, and manage a DNS server. In addition to the actual details of running a DNS server, we provide a detailed background on how DNS works and several troubleshooting tips, tricks, and tools.

From DNS, we move on to Chapter 17, "File Transfer Protocol (FTP)," which covers the installation and care of FTP servers. Like the DNS chapter, this chapter also includes a background on FTP itself and some notes on its evolution.

Chapter 18, "Apache Web Server," moves on to what may be considered one of the most popular uses of Linux today: running a web server with the popular Apache software. This chapter covers the information necessary to install, configure, and manage the Apache web server.

Chapter 19, "Simple Mail Transfer Protocol (SMTP)," and Chapter 20, "Post Office Protocol and Internet Mail Access Protocol (POP and IMAP)," dive into e-mail through the setup and configuration of SMTP, POP, and IMAP servers.

We cover the information needed to configure all three, as well as show how they interact with one another. We have chosen to cover the Postfix SMTP server instead of the classic Sendmail server, because Postfix provides a more flexible server with a better security record.

Chapter 21, "Voice over Internet Protocol (VoIP)," is an all new contribution and addition to this edition. A lot of work and effort went into brewing and distilling the very extensive topic of VoIP into just over 50 pages of this chapter. We think it was well worth it, and at the end of this chapter we build a simple VoIP-based PBX based on Asterisk software that can easily be extended to replace commercial-grade PBX solutions, make phone calls among local extensions, or interface with third-party VoIP providers to interface with the rest of the world.

Part IV ends with Chapter 22, "Secure Shell (SSH)." Knowing how to set up and manage the SSH service is useful in almost any server environment—regardless of the server's primary function.

Part V: Intranet Services

Again, we define intranet services as those that are typically run behind a firewall for internal users and mostly for internal consumption only. Even in this environment, Linux has a lot to offer. Part V starts off with Chapter 23, "Network File System (NFS)." NFS has been around for over 26 years now and has evolved and grown to fit the needs of its users quite well. This chapter covers Linux's NFS server capabilities, including how to set up both clients and servers, as well as troubleshooting.

Chapter 24, "Samba," continues the idea of sharing disks and resources with coverage of the Samba service. Using Samba, administrators can share disks and printing facilities and provide authentication for Windows (and Linux) users without having to install any special client software. Thus, Linux can become an effective server, able to support and share resources between UNIX/Linux systems as well as Windows systems. If you are into that *sort* of thing, Samba can even be configured to serve as a drop-in replacement for full-fledged Active Directory Microsoft Windows servers! Including Chapter 25, "Distributed File Systems (DFS)," in Part V instead of Part IV came down to a coin toss, because DFS can be used/deployed in both Internet- and intranet-facing scenarios. DFS solutions are especially important and relevant in today's cloud-centric world. Among the many DFS implementations available, we have selected to cover GlusterFS because of its ease of configuration and cross-distribution support.

In Chapter 26, "Network Information Service (NIS)," we talk about NIS, which is typically deployed alongside NFS servers to provide a central naming service for all users within a network. The chapter pays special attention to scaling issues and how you can make NIS work in an environment with a large user base.

We revisit directory services in Chapter 27, "Lightweight Directory Access Protocol (LDAP)," with coverage of LDAP and how administrators can use this standard service for providing a centralized user database (directory) for use among heterogeneous operating systems and also for managing tons of users.

Chapter 28, "Printing," takes a tour of the Linux printing subsystem. The printing subsystem, when combined with Samba, allows administrators to support seamless printing from Windows-based clients. The result is a powerful way of centralizing printing options for Linux, Windows, and even Mac OS X users on a single server.

Chapter 29, "Dynamic Host Configuration Protocol (DHCP)," covers another common use of Linux systems: DHCP servers. This chapter discusses how to deploy the Internet Systems Consortium (ISC) DHCP server, which offers a powerful array of features and access control options.

Then comes Chapter 30, "Virtualization." Over several months, we agonized and struggled over what title to use for this chapter, because we cover both virtualization and containers (containerization). We ended up naming the chapter simply "Virtualization" because we are not even sure if containerization is a word. (Our copy editor Bill McManus says it is a word in the *shipping* industry!) Virtualization is everywhere. It allows companies to consolidate services and hardware that previously required several dedicated bare-metal machines into much fewer bare-metal machines. We discuss the basic virtualization concepts and briefly cover some of the popular virtualization technologies in Linux. The chapter also covers the Kernel-based Virtual Machine (KVM) implementation in detail, with examples. The KVM concepts that we discuss will help prepare you for the brand-new goodies that we have in store for you in Appendix B. The tail end of Chapter 30 features containers. The concepts behind containers are *old* but *new* again. Like virtualization, containers are everywhere … and probably here to stay. We use the popular Docker platform for our container implementation and walk you through the process of how to deploy a web server container-style!

They say that all good things must come to an end, and that even applies to this book! The final chapter is Chapter 31, "Backups." Backups are arguably one of the most critical pieces of administration. Linux-based systems support several methods of providing backups that are easy to use and readily usable by tape drives and other media. The chapter discusses some of the methods and explains how they can be used as part of a backup schedule. In addition to the mechanics of backups, we discuss general backup design and how you can optimize your backup system.

Part VI: Appendixes

At the end of the book, we include some useful reference materials and real-world resources that you can use today and every day at work, at home, in the classroom, or in the lab.

Appendix A, "Creating a Linux Installer on Flash/USB Devices," details alternate and generic methods for creating an installation media on non-optical media, such as a USB flash drive, SD card, and so on.

Appendix B is another brand-new addition to this book. It covers one of the new features that we've added to this seventh edition: how to obtain and use a

purpose-built virtual machine (VM) image file that we created especially for you as an accompaniment to this book. Once you power up the VM, you'll see most of the commands, scripts, software packages, hacks, and servers/daemons that we discuss throughout this book. If you look carefully, you might even find some of our very own sys admin *mistakes* enshrined in this VM.

Typographic Conventions

We were very ambitious in aiming for this book to be the one-stop Linux administration resource for many of the mainstream Linux distros. But we didn't stop there! We also wanted to reflect the subtle nuances and idiosyncrasies that sometimes exist between mainstream distros. To help with this, we had to adopt some unique conventions throughout this book. Some of these conventions appear in code listings, commands, command output, and command prompts.

Commands that are meant to be typed out by the reader are in bold Courier font. For example:

```
command --options foobar
```

The output of commands is in plain Courier font:

```
This is a sample output
```

When we intend to reference or explain the output of commands that span multiple lines, we may sometimes number the output of such commands (also in Courier font). For example:

```
1. This is line 1 of a sample command output
2. This is line 2 of a sample command output
```

Some Linux commands and their options and output can be rather long. For these situations we place the backslash symbol (\) at the end of each line (except the final line) to indicate that the command spans multiple lines. For example, for a command with multiple options that spans multiple lines, we would show this as

```
command --option1 foobar1 --option2 foobar2 \
--option3 foobar3 --option4 foobar4 --option4 foobar4 \
--option5 foobar5
```

For distribution-independent commands, the command prompt will not have any particular distribution name. For example:

```
[root@server  ~]#
```

For commands that target a specific Linux distribution, the command prompt will incorporate the distribution name. For example, for a Fedora-based server, the command prompt will resemble

```
[root@fedora-server  ~]#
```

And for an Ubuntu-based server, the command prompt will resemble

```
root@ubuntu-server: ~#
```

Debian-based distros such as Ubuntu often do not directly use the superuser (*root*) or administrator account for administering the system. Regular user accounts with elevated privileges are used for this purpose. To achieve this, most commands are often prefixed with the **sudo** command on such platforms. Consider a regular user named "*master*" with **sudo** privileges who wants to run a restricted command. The command prompt and command prefixed with **sudo** will resemble

```
master@ubuntu-server:~$   sudo   command   --options
```

For projects or exercises that are meant to be run from a client system, the command prompt might appear as

```
user@client  ~$
```

For projects or exercises that are best suited for multi-server scenarios, when you absolutely need more than one server, the command prompt will change to reflect the steps or commands to be executed on the particular server. For example, the prompt for the first server (Server-A) might resemble

```
[root@server-A  ~]#
```

and the prompt for the second server (Server-B) might resemble

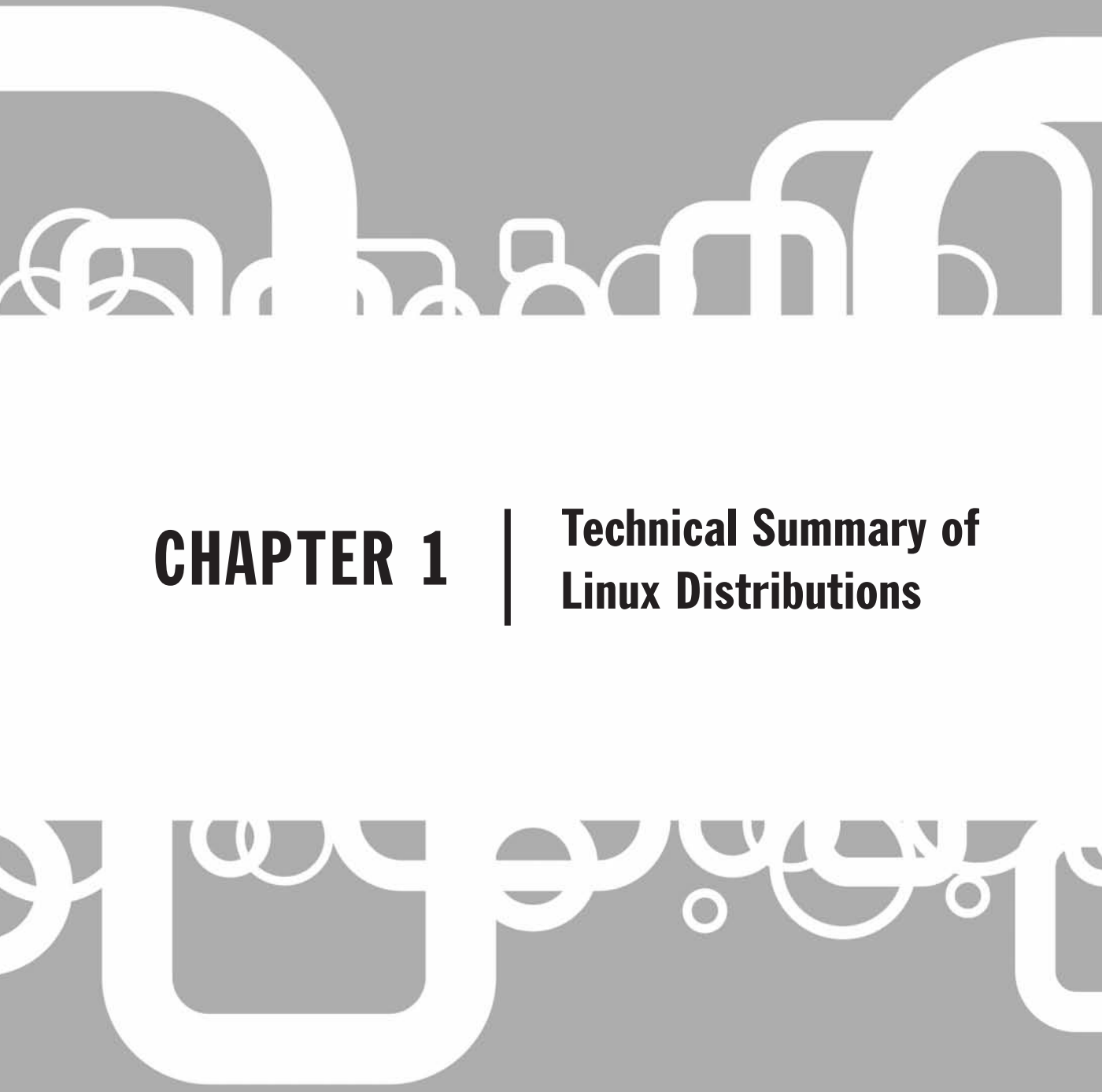
```
[root@server-B  ~]#
```

Updates and Feedback

Although we hope that we've published a book with no errors, we have set up an errata list for this book at www.linuxserverexperts.com. If you find any errors, we welcome your submissions for errata updates. We also welcome your feedback and comments. Unfortunately, our day jobs prevent us from answering detailed questions, so if you're looking for help on a specific issue, you may find one of the many online communities a useful resource. However, if you have two cents to share about the book, we welcome your thoughts. You can send us an e-mail to feedback@linuxserverexperts.com.

PART I | Introduction, Installation, and Software Management

This page intentionally left blank



CHAPTER 1 | Technical Summary of Linux Distributions

Linux has hit the mainstream. Hardly a day goes by without a mention of Linux (or open source software) in widely read and viewed print or digital media. What was only a hacker's toy many eons ago has grown up tremendously and is well known for its stability, performance, and extensibility.

If you need more proof concerning Linux's penetration, just pay attention to the frequency with which "Linux" is listed as a *desirable* and *must have* skill for technology-related job postings of Fortune 500 companies, small to medium-sized businesses, tech start-ups, and government, research, and entertainment industry jobs—to mention a few. The skills of good Linux system administrators and engineers are highly desirable!

With the innovations that are taking place in different open source projects (such as K Desktop Environment, GNOME, Unity, LibreOffice, Android, Apache, Samba, Mozilla, and so on), Linux has made serious inroads into consumer desktop, laptop, tablet, and mobile markets. This chapter looks at some of the core server-side technologies as they are implemented in the Linux (open source) world and in the Microsoft Windows Server world (possibly the platform you are more familiar with). But before delving into any technicalities, this chapter briefly discusses some important underlying concepts and ideas that form the genetic makeup of Linux and Free and Open Source Software (FOSS).

Linux: The Operating System

Some people (mis)understand Linux to be an entire software suite of developer tools, editors, graphical user interfaces (GUIs), networking tools, and so forth. More formally and correctly, such software *collectively* is called a *distribution,* or *distro.* The distro is the entire software suite that makes Linux useful.

So if we consider a distribution everything you need for Linux, what then *is* Linux exactly? Linux itself is the core of the operating system: the *kernel.* The kernel is the program acting as chief of operations. It is responsible for starting and stopping other programs (such as text editors, web browsers, services, and so on), handling requests for memory, accessing disks, and managing network connections. The complete list of kernel activities could easily fill a chapter in itself, and, in fact, several books documenting the kernel's internal functions have been written.

The kernel is a nontrivial program. It is also what puts the Linux badge on all the numerous Linux distributions. All distributions use essentially the same kernel, so the fundamental behavior of all Linux distributions is the same.

You've most likely heard of the Linux distributions named Red Hat Enterprise Linux (RHEL), Fedora, Debian, Mageia, Ubuntu, Mint, openSUSE, CentOS, Arch, Chrome OS, Slackware, and so on, which have received a great deal of press.

Linux distributions can be broadly categorized into two groups. The first category includes the purely commercial distros, and the second includes the noncommercial distros. The commercial distros generally offer support for their distribution—at a cost. The commercial distros also tend to have a longer release life cycle. Examples of commercial flavors of Linux-based distros are RHEL and SUSE Linux Enterprise (SLE).

The noncommercial distros, on the other hand, are free. These distros try to adhere to the original spirit of the open source software movement. They are mostly community supported and maintained—the community consists of the users and developers. The community support and enthusiasm can sometimes supersede that provided by the commercial offerings.

Several of the so-called noncommercial distros also have the backing and support of their commercial counterparts. The companies that offer the purely commercial flavors have vested interests in making sure that free distros exist. Some of the companies use the free distros as the proofing and testing ground for software that ends up in the commercial spins. Examples of noncommercial flavors of Linux-based distros are Fedora, openSUSE, Ubuntu, Linux Mint, Gentoo, and Debian. Linux distros such as Gentoo might be less well known and have not reached the same scale of popularity as Fedora, openSUSE, and others, but they are out there and in active use by their respective (and dedicated) communities.

What's interesting about the commercial Linux distributions is that most of the programs with which they ship were not written by the companies themselves. Rather, other people have released their programs with licenses, allowing their redistribution with source code. By and large, these programs are also available on other variants of UNIX, and some of them are becoming available under Windows as well. The makers of the distribution simply bundle them into one convenient and cohesive package that's easy to install. In addition to bundling existing software, several of the distribution makers also develop value-added tools that make their distribution easier to administer or compatible with more hardware, but the software that they ship is generally written by others. To meet certain regulatory requirements, some commercial distros try to incorporate/implement more specific security requirements that the FOSS community might not care about but that some institutions/corporations do care about.

Open Source Software and GNU: Overview

In the early 1980s, Richard Matthew Stallman began a movement within the software industry. He preached (and still does) that software should be free. Note that by *free*, he doesn't mean in terms of price, but rather free in the same sense as *freedom* or *libre.* This means shipping not just a product, but the entire source code as well. To clarify the meaning of free software, Stallman was once famously quoted as saying:

> *"Free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer."*

Stallman's position was, somewhat ironically, a return to classic computing, when software was freely shared among hobbyists on small computers and provided as part of the hardware by mainframe and minicomputer vendors. It was not until the late

1960s that IBM considered selling application software. Through the 1950s and most of the 1960s, IBM considered software as merely a tool for enabling the sale of hardware.

This return to *openness* was a wild departure from the early 1980s convention of selling prepackaged software, but Stallman's concept of open source software was in line with the initial distributions of UNIX from Bell Labs. Early UNIX systems did contain full source code. Yet by the late 1970s, source code was typically removed from UNIX distributions and could be acquired only by paying large sums of money to AT&T (now SBC). The Berkeley Software Distribution (BSD) maintained a free version, but its commercial counterpart, BSDi, had to deal with many lawsuits from AT&T until it could be proved that nothing in the BSD kernel came from AT&T.

Kernel Differences

Each company that sells a Linux distribution of its own will be quick to tell you that its kernel is better than the others. How can a company make this claim? The answer comes from the fact that each company maintains its own patch set. To make sure that the kernels largely stay in sync, most companies do adopt patches that are posted on www.kernel.org, the "Linux Kernel Archives." Vendors, however, typically do not track the release of every single kernel version that is released onto www.kernel.org. Instead, they take a foundation, apply their custom patches to it, run the kernel through their quality assurance (QA) process, and then take it to production. This helps organizations have confidence that their kernels have been sufficiently baked, thus mitigating any perceived risk of running open source–based operating systems.

The only exception to this rule revolves around security issues. If a security issue is found with a version of the Linux kernel, vendors are quick to adopt the necessary patches to fix the problem immediately. A new release of the kernel with the fixes is often made within a short time (commonly less than 24 hours) so that administrators who install it can be sure their installations are secure. Thankfully, exploits against the kernel itself are rare.

So if each vendor maintains its own patch set, what exactly is it patching? This answer varies from vendor to vendor, depending on each vendor's target market. Red Hat, for instance, is largely focused on providing enterprise-grade reliability and solid efficiency for application servers. This might be different from the mission of the Fedora team, which is more interested in trying new technologies quickly, and even more different from the approach of a vendor that is trying to put together a desktop-oriented or multimedia-focused Linux system.

What separates one distribution from the next are the value-added tools that come with each one. Asking, "Which distribution is better?" is much like asking, "Which is better, Coke or Pepsi?" Almost all colas have the same basic

ingredients—carbonated water, caffeine, and high-fructose corn syrup—thereby giving the similar effect of quenching thirst and bringing on a small caffeine-and-sugar buzz. In the end, it's a question of requirements: Do you need commercial support? Did your application vendor recommend one distribution over another? Does the software (package) updating infrastructure suit your site's administrative style better than another distribution? When you review your requirements, you'll find that there is likely a distribution that is geared toward your exact needs.

The idea of giving away source code is a simple one: A user of the software should never be forced to deal with a developer who might or might not support that user's intentions for the software. The user should never have to wait for bug fixes to be published. More important, code developed under the scrutiny of other programmers is typically of higher quality than code written behind locked doors. One of the great benefits of open source software comes from the users themselves: Should they need a new feature, they can add it to the original program and then contribute it back to the source so that everyone else can benefit from it.

This line of thinking sprung a desire to release a complete UNIX-like system (Linux) to the public, free of license restrictions. Of course, before you can build any operating system, you need to build tools. And this is how the GNU project and its namesake license were born.

NOTE GNU stands for GNU's Not UNIX—recursive acronyms are part of hacker humor. If you don't understand why it's funny, don't worry. You're still in the majority.

The GNU Public License

An important thing to emerge from the GNU project is the GNU Public License (GPL). This license explicitly states that the software being released is free and that no one can ever take away these freedoms. It is acceptable to take the software and resell it, even for a profit; however, in this resale, the seller must release the full source code, including any changes. Because the resold package remains under the GPL, the package can be distributed for free and resold yet again by anyone else for a profit. Of primary importance is the liability clause: The programmers are not liable for any damages caused by their software.

It should be noted that the GPL is not the only license used by open source software developers (although it is arguably the most popular). Other licenses, such as BSD and Apache, have similar liability clauses but differ in terms of their redistribution. For instance, the BSD license allows people to make changes to the code and ship those changes without having to disclose the added code. (Whereas the GPL requires that the added code is shipped.) For more information about other open source licenses, check out www.opensource.org.

Historical Footnote

Many, many moons ago, Red Hat started a commercial offering of its erstwhile free product (Red Hat Linux). The commercial release gained steam with the Red Hat Enterprise Linux (RHEL) series. Because the foundation for RHEL is GPL, individuals interested in maintaining a free version of Red Hat's distribution have been able to do so. Furthermore, as an outreach to the community, Red Hat created the Fedora Project, which is considered the testing grounds for new software before it is adopted by the RHEL team. The Fedora Project is freely distributed and can be downloaded from http://fedoraproject.org or https://getfedora.org.

Upstream and Downstream

To help you understand the concept of upstream and downstream components, let's start with an analogy. Picture, if you will, a pizza with all your favorite toppings.

The pizza is put together and baked by a local *pizza shop.* Several things go into making a great pizza—cheeses, vegetables, flour (dough), herbs, meats, to mention a few. The pizza shop will often make some of these ingredients in-house and rely on other businesses to supply other ingredients. The pizza shop will also be tasked with assembling the ingredients into a complete finished pizza.

Let's consider one of the most common pizza ingredients—cheese. The cheese is made by a *cheesemaker* who makes her cheese for many other industries or applications, including the pizza shop. The cheesemaker is pretty set in her ways and has very strong opinions about how her product should be paired with other food stuffs (wine, crackers, bread, vegetables, and so on). The pizza shop owners, on the other hand, do not care about other food stuffs—they care only about making a great pizza. Sometimes the cheesemaker and the pizza shop owners will bump heads because of differences in opinion and objectives. And at other times they will be in agreement and cooperate beautifully. Ultimately (and sometimes unbeknown to them), the pizza shop owners and cheesemaker care about the same thing: producing the best product that they can.

The pizza shop in our analogy here represents the Linux distributions vendors/projects (Fedora, Debian, RHEL, openSUSE, and so on). The cheesemaker represents the different software project maintainers that provide the important programs and tools (such as the Bourne Again Shell [Bash], GNU Image Manipulation Program [GIMP], GNOME, KDE, Nmap, and GNU Compiler Collection [GCC]) that are packaged together to make a complete distribution (pizza). The Linux distribution vendors are referred to as the *downstream* component of the open source food chain; the maintainers of the accompanying different software projects are referred to as the *upstream* component.

Standards

One argument you hear regularly against Linux is that too many different distributions exist, and that by having multiple distributions, fragmentation occurs. The argument opines that this fragmentation will eventually lead to different versions of incompatible distros.

This is, without a doubt, complete nonsense that plays on "FUD" (fear, uncertainty, and doubt). These types of arguments usually stem from a misunderstanding of the kernel and distributions.

Ever since becoming so mainstream, the Linux community understood that it needed a formal method and standardization process for how certain things should be done among the numerous Linux spins. As a result, two major standards are actively being worked on.

The *Filesystem Hierarchy Standard (FHS)* is an attempt by many of the Linux distributions to standardize on a directory layout so that developers have an easy time making sure their applications work across multiple distributions without difficulty. As of this writing, several major Linux distributions have become completely compliant with this standard.

The *Linux Standard Base (LSB)* specification is a standards group that specifies what a Linux distribution should have in terms of libraries and tools.

A developer who assumes that a Linux machine complies only with LSB and FHS is almost guaranteed to have an application that will work with all compliant Linux installations. All of the major distributors have joined these standards groups. This should ensure that all desktop distributions will have a certain amount of commonality on which a developer can rely.

From a system administrator's point of view, these standards are interesting but not crucial to administering a Linux environment. However, it never hurts to learn more about both. For more information on the FHS, go to their web site at www.pathname.com/fhs. To find out more about LSB, check out www.linuxfoundation.org/collaborate/workgroups/lsb.

The Advantages of Open Source Software

If the GPL seems like a bad idea from the standpoint of commercialism, consider the surge of successful open source software projects—they are indicative of a system that does indeed work. This success has evolved for two reasons. First, as mentioned earlier, errors in the code itself are far more likely to be caught and quickly fixed under the watchful eyes of peers. Second, under the GPL system, programmers can release code without the fear of being sued. Without that protection, people might not feel as comfortable to release their code for public consumption.