EXAM CRAM

CompTIA®

Linux+

XK0-005


Cram Sheet


Practice Tests


Exam Alerts

WILLIAM "BO" ROTHWELL



EXAM CRAM

CompTIA® Linux+® XK0-005 Exam Cram

William “Bo” Rothwell

CompTIA® Linux+® XK0-005 Exam Cram

ISBN-13: 978-0-13-789855-8

ISBN-10: 0-13-789855-X

Library of Congress Control Number: 2022910969

ScoutAutomatedPrintCode

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Pearson IT Certification cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

This book is designed to provide information about the CompTIA® Linux+® (XK0-005) certification. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact

governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact

intlcs@pearson.com.

Editor-in-Chief
Mark L. Taub

Director, ITP Product Management
Brett Bartow

Executive Editor
Nancy Davis

Development Editor
Christopher A. Cleveland

Managing Editor
Sandra Schroeder

Project Editor
Mandie Frank

Copy Editor
Kitty Wilson

Indexer
Erika Millen

Proofreader
Donna E. Mulder

Technical Editor
Casey Boyles

Publishing Coordinator
Cindy Teeters

Designer
Chuti Prasertsith

Compositor
codeMantra

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where

- Everyone has an equitable and lifelong opportunity to succeed through learning
- Our educational products and services are inclusive and represent the rich diversity of learners
- Our educational content accurately reflects the histories and experiences of the learners we serve
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview)

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

Contents at a Glance

Part IV: Troubleshooting

Table of Contents

Part III: Scripting, Containers, and Automation

Figure Credits

Figure	Credit
Figures 1.3, 1.4, 2.2	GNU Project
Figures 2.3, 4.2, 5.2, 6.2, 14.1–14.8, 19.1, 19.2, 20.1, 20.2	Linux Kernel Organization, Inc
Figure 5.1	Wireshark
Figure 11.1	Mozilla.org

About the Author

At the impressionable age of 14, **William "Bo" Rothwell** crossed paths with a TRS-80 Micro Computer System (affectionately known as a "Trash 80"). Soon after the adults responsible for Bo made the mistake of leaving him alone with the TSR-80, he dismantled it and held his first computer class, showing his friends what made this "computer thing" work.

Since that experience, Bo's passion for understanding how computers work and sharing this knowledge with others has resulted in a rewarding career in IT training. His experience includes Linux, Unix, IT security, DevOps, cloud technologies, and programming languages such as Perl, Python, Tcl, and BASH. He is the founder and lead instructor of One Course Source, an IT training organization.

Dedication

As I close out what will become my 14th book in print (and my 10th with Pearson Publishing), I find myself writing YAD (yet another dedication). I honestly didn't know who I was going to dedicate this book to until just yesterday, when my family had to make one of the most difficult decisions of my life. We needed to end the suffering of our amazing, faithful, and lovable dog, Midnight, a black lab/golden retriever mix.

I was reminded, in a very emotionally painful way, how our furry family members mean so much to us. Midnight brought so much joy and happiness to our family and asked only simple things in return: affection, the opportunity to be close to the members of his pack, and, of course, treats.

He made my world a bit brighter, and while the world is a bit dimmer today, I know that my memory of him will forever enrich my life.

I will miss you, Midnight.

Acknowledgments

To everyone at Pearson who helped make this book come to life, I thank you. I know that this is a team effort, and I appreciate everyone's hard work.

Special thanks go to Nancy, Chris, and Casey for helping me complete this book ahead of schedule!

About the Technical Reviewer

Casey Boyles started working in the IT field more than 30 years ago and quickly moved into systems automation, distributed applications, and database development. Casey later moved into technical training and course development, where he specializes in Layer 0–7 software development, database architecture, systems security, telecommunications, and cloud computing. Casey typically spends his time smoking cigars while "reading stuff and writing stuff."

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: community@informit.com

Reader Services

Register your copy of *CompTIA Linux+ XK0-005 Exam Cram* at www.pearsonitcertification.com for convenient access to downloads, updates, and corrections as they become available. To start the registration process, go to www.pearsonitcertification.com/register and log in or create an account.* Enter the product ISBN **9780137898558** and click **Submit**. When the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box indicating that you would like to hear from us to receive exclusive discounts on future editions of this product.

Introduction

Welcome to *CompTIA Linux+ XK0-005 Exam Cram*. This book prepares you for the CompTIA Linux+ XK0-005 certification exam. Imagine that you are at a testing center and have just been handed the passing scores for this exam. The goal of this book is to make that scenario a reality. My name is Bo Rothwell, and I am happy to have the opportunity to help you in this endeavor. Together, we can accomplish your goal to attain the CompTIA Linux+ certification.

Target Audience

The CompTIA Linux+ exam measures the necessary competencies for an entry-level Linux professional with the equivalent knowledge of at least 12 months of hands-on experience in the lab or field.

This book is for persons who have experience working with Linux operating systems and want to cram for the CompTIA Linux+ certification exam—*cram* being the key word.

Linux can be a challenging topic for individuals who are not used to command-line environments. If you don't already have a lot of experience running commands in Linux, I highly recommend trying out the commands presented in this book. Install Linux on a virtual machine and get to practicing!

This book focuses very specifically on the CompTIA Linux+ certification exam objectives. I point this out because you might consider exploring other topics if you want to become proficient. I avoided any non-testable topics because I didn't want to add any confusion as to what you need to study to pass the exam. You might find that some topics that are not exam-testable, like installing Linux and using man pages (to view documentation), will be useful for your understanding of the Linux operating system.

About the CompTIA Linux+ Certification

This book covers the CompTIA Linux+ XK0-005 exam, which you will need to pass to obtain the CompTIA Linux+ certification. This exam is administered by Pearson Vue and can be taken at a local test center or online.

Passing the certification exam proves that you have a solid understanding of the essentials of the Linux operating system, as well as associated Linux topics.

Before doing anything else, I recommend that you download the official CompTIA Linux+ objectives from CompTIA's website. The objectives are a comprehensive bulleted list of the concepts you should know for the exams. This book directly aligns with those objectives, and each chapter specifies the objective it covers.

For more information about how the Linux+ certification can help you in your career or to download the latest objectives, access CompTIA's Linux+ web page at https://www.comptia.org/certifications/linux.

About This Book

This book covers what you need to know to pass the CompTIA Linux+ exam. It does so in a concise way that allows you to memorize the facts quickly and efficiently.

We organized this book into four parts comprising 22 chapters, each chapter pertaining to a particular objective covered on the exams. Each part of the book matches up exactly with one of the four Linux+ exam domains.

A note about studying for the exam: The chapters in this book are in exactly the same order as the corresponding objectives on the Linux+ exam. This provides you with a very clear understanding of where to find content for a specific exam objective, but this does not necessarily mean that you should read the book from cover to cover. For example, Chapter 1, "Linux Fundamentals," does not cover "entry-level" Linux topics. The chapter title matches the Linux+ objective, but if you review the topics, you will discover that they are more "foundational" in nature, not the fundamental topics that an entry-level person would learn. So, where are these fundamental topics in the book? They start in Chapter 2, "Manage Files and Directories."

I mention this because if you are a novice Linux learner and are trying to learn Linux from the ground up using this book, you will likely become overwhelmed within the first chapter. With that said, this really isn't a "learn from the ground up book" but rather a book designed to fill in a bunch of gaps that Linux users often find they have when preparing for the Linux+ exam.

Chapter Format and Conventions

Every chapter of this book follows a standard structure and contains graphical clues about important information. Each chapter includes the following:

- **Opening topics list:** This list defines the CompTIA Linux+ objective covered in the chapter.

- **Topical coverage:** The heart of the chapter, this text explains the topics from a hands-on and theory-based standpoint. In-depth descriptions, tables, and figures are geared toward helping you build your knowledge so that you can pass the exam.
- **Cram Quiz questions:** At the end of each chapter is a brief quiz, along with answers and explanations. The quiz questions and ensuing explanations are meant to help you gauge your knowledge of the subjects you have just studied. If the answers to the questions don't come readily to you, consider reviewing individual topics or the entire chapter. You can also find the Cram Quiz questions on the book's companion web page, at www.pearsonitcertification.com.
- **ExamAlerts and Notes:** These are interspersed throughout the book. Watch out for them!

ExamAlert

This is what an ExamAlert looks like. ExamAlerts stress concepts, terms, hardware, software, or activities that are likely to relate to one or more questions on the exam.

Additional Elements

Beyond the chapters, we have provided some additional study aids for you:

- **CramSheet:** The tear-out CramSheet is located in the beginning of the book. It jams some of the most important facts you need to know for each exam into one small sheet, allowing for easy memorization. It is also available in PDF format on the companion web page. If you have an e-book version, the CramSheet might be located elsewhere in the e-book; run a search for the term "cramsheet," and you should be able to find it.
- **Online Practice Exams:** If you want more practice on the exam objectives, remember that you can access all of the Cram Quiz questions on the Pearson Test Prep software online. You can also create a custom exam, by objective, with the Online Practice Test. Note any objective you struggle with and go to that objective's material in the corresponding chapter. Download the Pearson Test Prep Software online at http://www.pearsonitcertification.com/content/downloads/pcpt/engine.zip.

To access the book's companion website and the software, simply follow these steps:

Step 1. Register your book by going to **PearsonITCertification.com/register** and entering the ISBN **9780137898558**.

Step 2. Answer the challenge questions.

Step 3. Go to your account page and click the **Registered Products** tab.

Step 4. Click the **Access Bonus Content** link under the product listing.

Step 5. Click the **Install Pearson Test Prep Desktop Version** link under the Practice Exams section of the page to download the software.

Step 6. After the software finishes downloading, unzip all the files on your computer.

Step 7. Double-click the application file to start the installation and follow the onscreen instructions to complete the registration.

Step 8. After the installation is complete, launch the application and click the **Activate Exam** button on the My Products tab.

Step 9. Click the **Activate a Product** button in the Activate Product Wizard.

Step 10. Enter the unique access code found on the card in the sleeve in the back of your book and click the **Activate** button.

Step 11. Click **Next** and then click **Finish** to download the exam data to your application.

Step 12. Start using the practice exams by selecting the product and clicking the **Open Exam** button to open the exam settings screen.

You can also use the online version of this software on any device with a browser and connectivity to the Internet including desktop machines, tablets, and smartphones. Follow the directions on the companion website for the book. Note that the offline and online versions will sync together, so saved exams and grade results recorded in one version will be available to you in the other as well.

The Hands-On Approach

As mentioned previously, hands-on experience is very important for understanding Linux. Before taking the exam, you should practice using each command that is listed in this book. Explore the different options that are provided in this book to gain a better understanding of each topic.

Use a virtual machine! It is possible that when you perform some of the administration tasks (partitioning, using firewalls, and so on), you could end up making the operating system unusable. If you use a virtual machine and mess up the original, you can just install a new one (or make use of a cool feature called a snapshot, which allows you to return your operating system to a previous state).

Goals for This Book

Clearly, the primary goal of this book is to prepare you to pass the Linux+ certification exam. With that goal in mind, I did my best to include all relevant exam topics, commands, and information in a very condensed format.

The secondary goal of this book is the help you broaden your understanding of Linux. The folks who developed the objectives for the Linux+ exam did an excellent job of including a wide variety of Linux-related topics. I've done my best to ensure that you have a good understanding of each of these topics, within the bounds of what is testable on the exam.

Linux is a truly remarkable topic, which includes a wide range of capabilities. After achieving your goal of passing the Linux+ exam, I highly encourage you to explore this topic further.

Good luck with the exam and please feel free to reach out to me on LinkedIn, at https://www.linkedin.com/in/bo-rothwell/.

I look forward to hearing about your journey toward passing the Linux+ exam!

—William "Bo" Rothwell

CHAPTER 1

Linux Fundamentals

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.1:** Summarize Linux fundamentals.

Welcome to the first chapter of the book, where you will learn about some of the fundamental features of Linux. In this chapter you will learn about the common locations where Linux files are stored by exploring the Filesystem Hierarchy Standard (FHS). You will also explore the boot process, including BIOS, UEFI, and GRUB2.

Later in this chapter you will learn about device types and how to perform a basic package compilation from source code. The chapter ends with coverage of storage concepts and commands that are used to list hardware information.

This chapter provides information on the following topics: the Filesystem Hierarchy Standard (FHS), the basic boot process, kernel panic, device types in **/dev**, basic package compilation from source, storage concepts, and hardware information.

Filesystem Hierarchy Standard (FHS)

The Filesystem Hierarchy Standard (FHS) defines where files and directories are supposed to be placed on Unix and Linux operating systems. Table 1.1 provides a summary of some of the most important locations.

TABLE 1.1 **FHS Locations**

Location	Description/Contents
/	The root or top-level directory
/bin	Critical binary executables
/boot	Files related to booting the system
/dev	Files that represent physical devices (See the section "Device Types in **/dev**," later in this chapter, for more details.)
/etc	Configuration files for the system
/home	Regular user home directories
/lib	Critical system libraries
/media	Mount points for removable media
/mnt	Temporary mounts
/opt	Optional software packages
/proc	Information related to kernel data and process data (in a virtual file-system, not a disk-based filesystem)
/root	Home directory for the root user account
/sbin	Critical system binary executables
/sys	Files that contain system-related information
/tmp	Temporary files
/usr	Many subdirectories that contain binary executables, libraries, and documentation
/usr/bin	Nonessential binary executables
/usr/lib	Libraries for the executables in the **/usr/bin** directory
/usr/sbin	Nonessential system binary executables
/usr/share	Data that is architecture independent
/var	Data that is variable (that is, that changes in size regularly)
/var/mail	Mail logs
/var/log	Spool data (such as print spools)
/var/tmp	Temporary files

ExamAlert

For the Linux+ XK0-005 exam, you should know where files are stored in Linux. Review Table 1.1 prior to taking the exam.

Basic Boot Process

A *bootloader* is a piece of software that is designed to handle the initial booting of the operating system (OS). Figure 1.1 provides an overview of the boot process and the bootloader's place in this process.

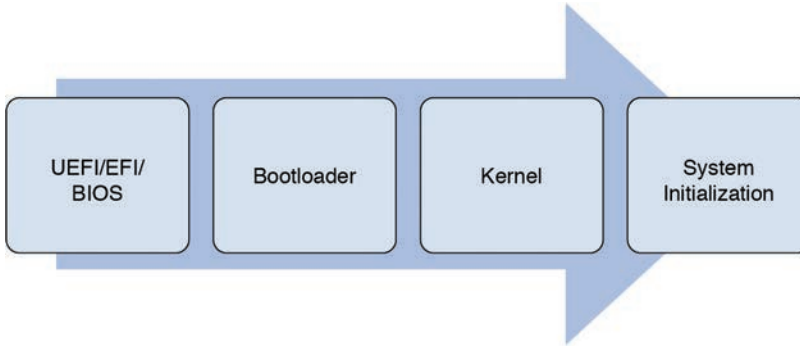

FIGURE 1.1 **Overview of the Boot Process**

UEFI/EFI/BIOS performs sanity checks and then loads the bootloader. See the "Basic Input/Output System (BIOS)/Unified Extensible Firmware Interface (UEFI)" section, later in this chapter, for more details.

The standard Linux bootloader is the Grand Unified Bootloader (GRUB or GRUB2). It is responsible for loading the kernel and associated kernel modules (or *libraries*) stored in a file referred to as the **initramfs** file.

The **initramfs** file contains a mini-root filesystem that has the kernel modules necessary when the system is booting. It is located in the **/boot** filesystem, and there is a unique **initramfs** file for each kernel. The **initramfs** file is created by using the **mkinitrd** command (see the "**mkinitrd**" section, later in this chapter, for more information).

The kernel is loaded from the hard disk, performs some critical boot tasks, and then passes control of the boot process to the system initialization software.

The three different system initialization systems in Linux are SysVinit (the oldest), Upstart, and Systemd (currently the most widely used). The system initialization is responsible for starting system services.

Basic Input/Output System (BIOS)/ Unified Extensible Firmware Interface (UEFI)

Basic input/output system (BIOS), Unified Extensible Firmware Interface (UEFI), and Extensible Firmware Interface (EFI) are all similar in that they are used to provide connections between a system's firmware and the operating system. These programs are provided by the system's manufacturer and are able to start the boot process.

BIOS is only mentioned here in passing. It is older software that has not been officially supported since 2020. However, many UEFI and EFI systems are often referred to as "BIOS," and it is important that you understand this.

UEFI is the successor to EFI and considered the standard in most modern systems.

For the Linux+ XK0-005 exam, you should be aware that UEFI/EFI is the software that starts the boot process. It is the component that starts the bootloader. In addition, it is configurable; for example, you can specify which devices (hard disk, CD/DVD, and so on) to boot from and in which order to attempt to find a bootloader on these devices.

Commands

The sections that follow focus on the commands related to boot software.

mkinitrd

The **initrd** file is created by the **mkinitrd** command, which in turn calls the **dracut** utility:

```
[root@localhost ~]# mkinitrd /boot/initrd-5.17.4.x86_64.img 5.17.4
```

The first argument to the **mkinitrd** command is the name of the **initrd** file that you want to create. The second argument is the version of the kernel.

Note that you rarely use the **dracut** utility directly; however, it is listed as a Linux+ XK0-005 exam objective, so be aware that **mkinitrd** executes the **dracut** command behind the scenes.

See the section "**initrd.img**," later in this chapter, for information on how this file is generated.

grub2-install

Typically the bootloader is installed during the boot process, but it is possible that the bootloader could become corrupt and need to be reinstalled. To install the bootloader, execute the **grub-install** command and provide the device where you want to install GRUB. For example, the following command installs GRUB on the first SATA hard drive:

```
[root@localhost ~]# grub2-install /dev/sda
```

grub2-mkconfig

grub2-mkconfig, which is used only for GRUB2, generates GRUB2 configuration files from the user-editable files located in the **/etc** directory structure. This command converts data from the **/etc/default/grub** file and the files in the **/etc/grub.d** directory into the GRUB2 configuration file (either **/boot/grub/grub.cfg** or **/boot/grub/menu.lst**).

Figure 1.2 provides a visual example.

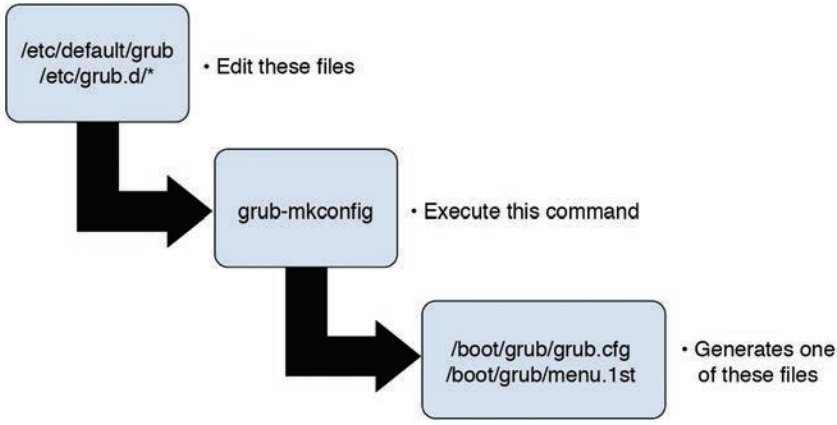

FIGURE 1.2 **The grub2-mkconfig Command**

> **Note**
>
> On some systems, the command is **grub-mkconfig**.

grub2-update

The **grub2-update** command provides another way of running the **grub2-mkconfig** utility. It exists mostly for backward compatibility to some systems that utilized this command to update the GRUB2 configuration files. By default it runs the command **grub-mkconfig -o /boot/grub/grub.cfg**. See the "**grub2-mkconfig**" section, earlier in this chapter, for details about that command.

dracut

Refer to the "**mkinitrd**" section, earlier in this chapter, for more information about **dracut**.

initrd.img

Typically the Linux kernel is configured with few kernel modules enabled by default. Additional modules are normally required during the boot process before the filesystems are mounted. These additional modules are stored within a compressed file called **initrd.img**. See the "**mkinitrd**" section, earlier in this chapter, for information on how this file is generated.

vmlinuz

The **vmlinuz** file is stored in the **/boot** directory. Each version of the kernel has a different **vmlinuz** file, typically with a name that includes the version of the kernel. Here is an example:

```
vmlinuz-4.17.8-200.fc32.x86_64
```

Grand Unified Bootloader Version 2 (GRUB2)

The Grand Unified Bootloader (GRUB), also called Legacy GRUB, is an older bootloader that is rarely used on modern Linux systems. Most of the configuration files and commands on the Linux+ XK0-005 exam focus on GRUB2, which is an improved version of GRUB.

GRUB2 is designed as a replacement for Legacy GRUB. There are several differences between the two, including the following:

- They use different configuration files.

- GRUB2 supports more devices to boot from, including LVM (Logical Volume Management) and software RAID devices.
- GRUB2 supports UEFI and EFI. See the section "Basic Input/Output System (BIOS)/Unified Extensible Firmware Interface (UEFI)," earlier in this chapter, for more details.

Expect Linux+ XK0-005 exam questions to focus on GRUB2, as Legacy GRUB is rarely used in modern Linux distributions.

Boot Sources

GRUB2 allows you to boot from different media. This section focuses on which media you can boot from as well as how to boot from the different media sources.

During the boot process, you can interact with the bootloader. This is normally useful for the following reasons:

- To boot to an alternative stanza
- To modify the existing boot parameters

This interaction starts with the boot menu screen, as shown in Figure 1.3.

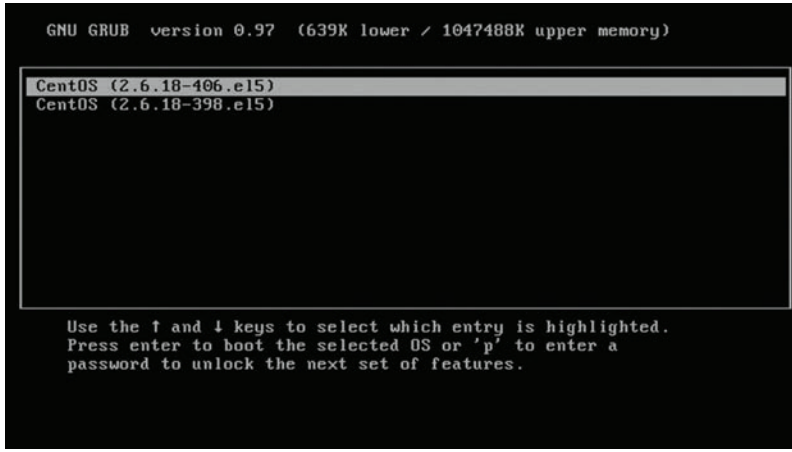

FIGURE 1.3 **The GRUB Boot Menu Screen**

Table 1.2 describes the commands available on the GRUB boot menu screen.

TABLE 1.2 **Commands Available on the GRUB Boot Menu Screen**

Command	Description
Arrow keys	Used to select a stanza.
e	Used to edit the currently selected stanza.
c	Used to enter a GRUB command prompt.
p	Only visible when a password is required to edit a stanza; use **p** to enter the required password.

If you edit a stanza, a new screen with different menu options is provided (see Figure 1.4).

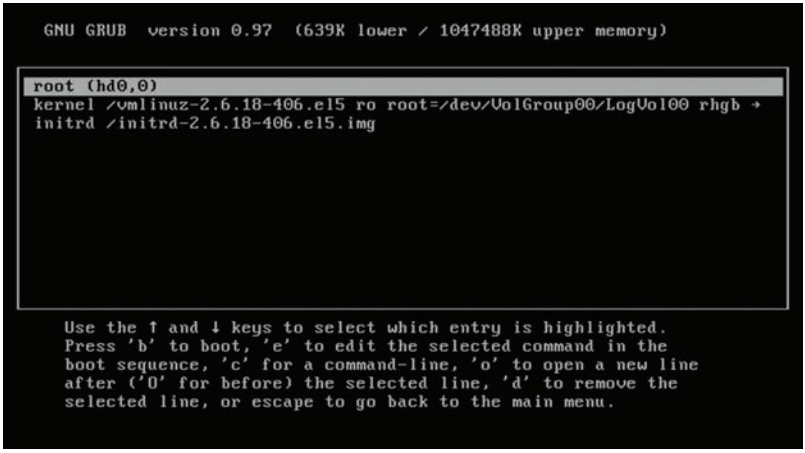

FIGURE 1.4 The GRUB Stanza-Editing Screen

Table 1.3 describes the commands available on the GRUB stanza-editing screen.

TABLE 1.3 **Commands Available on the GRUB Boot Stanza-Editing Screen**

Command	Description
Arrow keys	Used to select a stanza.
e	Used to edit the currently selected line.
c	Used to enter a GRUB command prompt.
o	Used to open (create) a new line below the current line.
O	Used to open (create) a new line above the current line.
d	Used to remove the selected line.

Command	Description
b	Used to boot the current stanza.
[ESC]	Returns you to the main menu.

The rest of this section describes the different boot sources that GRUB2 can boot from.

Preboot eXecution Environment (PXE)

Preboot eXecution Environment (PXE) allows you to boot a system across the network, assuming that a boot server has been created on the network. PXE uses a Dynamic Host Configuration Protocol (DHCP) server to obtain network configuration information, such as an IP address and subnet address.

The boot server listens for PXE boot requests and then provides an operating system to the client system. Typically this operating system calls the client machine to perform an installation, so further PXE boots are not required.

Note that PXE boots are initiated from BIOS/UEFI/EFI software.

Booting from ISO/Universal Serial Bus (USB)

There are several Live Linux distributions that allow you to boot directly from a CD, DVD, or USB device. This technique is referred to as "booting from an ISO" because the file format used to store the operating system on this media is called an *ISO image*.

There are several advantages to and reasons for booting from an ISO:

- The system might be a thin client with no hard drive.
- Booting from a security-based ISO image can be helpful in resolving issues (virus, worms, or other security compromises) on the host OS.
- Some Live Linux distributions can fix problems with booting the host OS.
- An ISO image can also be used to install a new distribution on the system.

In most cases, booting from an ISO image requires instructing BIOS/UEFI/EFI to boot from the drive that contains the ISO media. However, it is possible to configure GRUB to boot from ISO media as well (although this tends to be rarely done).

Kernel Panic

A *kernel panic* occurs when something goes wrong with the kernel and the system crashes. Typically when this happens, data is stored using a feature called **kdump**. A kernel expert can view this data to determine the cause of the crash.

ExamAlert

Using a **kdump** file is a specialized skill and beyond the scope of the Linux+ XK0-005 exam.

Device Types in /dev

The **/dev** filesystem contains device files, which are used to access physical devices (such as hard drives, keyboards, and CPUs) and virtual devices (such as LVM devices, pseudo-terminals, and software RAID devices). The **/dev** filesystem is memory based, not stored on the hard drive.

Table 1.4 describes the key files in **/dev**.

TABLE 1.4 **Key Files in /dev**

File	Description
/dev/sd*	Devices that begin with **sd** in the **/dev** directory are either SATA, SCSI, or USB devices. The device name **/dev/sda** refers to the first device, **/dev/sdb** refers to the second device, and so on. If a device has partitions, they are numbered starting with 1. For example, **/dev/sda1** is the first partition of the first SATA, SCSI, or USB device.
/dev/hd*	Devices that begin with **hd** in the **/dev** directory are IDE-based devices. The device name **/dev/hda** refers to the first device, **/dev/hdb** refers to the second device, and so on. If a device has partitions, they are numbered starting with 1. For example, **/dev/hda1** is the first partition of the first IDE-based device.
/dev/cdrom	This is a symbolic link that points to the first CD-ROM on the system.
/dev/dm*	Devices that begin with **dm** in the **/dev** directory are either software RAID or LVM devices. The device name **/dev/dm-0** refers to the first device, **/dev/dm-1** refers to the second device, and so on.
/dev/tty*	Devices that begin with **tty** in the **/dev** directory are terminal devices. The device name **/dev/tty0** refers to the first device, **/dev/tty1** refers to the second device, and so on.

ExamAlert

Be prepared for Linux+ XK0-005 exam questions related to specific device files described in this section.

Block Devices

A *block device* is any device that is designed to read and write data in chunks (that is, blocks). Block devices are typically storage devices, like USB drives, hard drives, CD-ROMs, and DVDs.

Character Devices

A *character device* is a device that is designed to read and write data in single bits (that is, single characters). An example of a character device is a keyboard, which sends one character to the system at a time.

Special Character Devices

Special character devices are device files that don't represent real physical devices. This includes the **/dev/null**, **/dev/zero**, and **/dev/urandom** files, as described in the sections that follow.

/dev/null

In some cases, you may not want to see either stdout or stderr of a command. For example, consider the following **find** command:

```
[root@OCS ~]$ find /etc -name "hosts"
find: '/etc/named': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
find: '/etc/polkit-1/rules.d': Permission denied
find: '/etc/pki/rsyslog': Permission denied
find: '/etc/pki/CA/private': Permission denied
find: '/etc/sudoers.d': Permission denied
find: '/etc/grub.d': Permission denied
find: '/etc/phpMyAdmin': Permission denied
/etc/hosts
```

```
find: '/etc/selinux/targeted/modules/active': Permission denied
find: '/etc/webmin/lpadmin': Permission denied
find: '/etc/webmin/iscsi-target': Permission denied
```

Notice the large number of "Permission denied" messages. These messages result from not having permissions to view the contents of specific directories that are located in the search path. You often don't care about such messages and would rather not see them. In such cases, you can redirect these messages to the **/dev/null** device file. This file is called the "bit bucket" and acts as a trash can that never needs to be emptied. Anything sent to the **/dev/null** device is immediately discarded.

Because the error messages in the previous **find** command were sent to stderr, they can be discarded by using the following command:

```
[root@OCS ~]$ find /etc -name "hosts" 2> /dev/null
/etc/hosts
```

See Chapter 13, "Create Simple Shell Scripts to Automate Common Tasks," for details regarding the **2>** symbol, stdout, and stderr.

/dev/zero

The **/dev/zero** file is a special file in Linux that returns null characters. It is often used with utilities like the **dd** command to create large files. See the "**dd**" section in Chapter 2, "Manage Files and Directories," for more details about the **dd** command and how it uses the **/dev/zero** file.

/dev/urandom

The Linux kernel has a feature that can be used to generate random numbers. This feature can be accessed by reading content from the **/dev/urandom** file. This file returns random numbers, which can be useful in software development as well as with some tools that need to create unique values, such as encryption tools.

ExamAlert

Using **/dev/urandom** is beyond the scope of the Linux+ XK0-005 exam.

Basic Package Compilation from Source

Build tools are used to create software that will be used by others. This section explores some of the most useful build tools.

./configure

After you download the source code of a software package, you need to know how to build it and install it. In a real-world situation, you would probably modify the source code first, but that normally requires programming skills that are beyond the scope of this book and the Linux+ XK0-005 exam.

The first step in building the source code is to execute the **configure** script. This may be directly in the top-level directory of the source code, but you may need to look for it, as in the following output:

```
[root@OCS unix]# pwd
/tmp/source/zip/unix
[root@OCS unix]# ls
configure  Makefile  osdep.h  Packaging  README.OS390  unix.c  zipup.h
```

You need to change the permissions of the configure file to make it executable and then run the script as shown in the following example:

```
[root@OCS unix]# su - bo
[bo@OCS unix]$ chmod u+x configure
[bo@OCS unix]$ ./configure
Check C compiler type (optimization options)
  GNU C (-O3)
Check bzip2 support
  Check for bzip2 in bzip2 directory
  Check if OS already has bzip2 library installed
-- Either bzlib.h or libbz2.a not found - no bzip2
Check for the C preprocessor
Check if we can use asm code
Check for ANSI options
Check for prototypes
```

```
Check the handling of const

Check for time_t

Check for size_t

Check for off_t

Check size of UIDs and GIDs

(Now zip stores variable size UIDs/GIDs using a new extra field. This
tests if this OS uses 16-bit UIDs/GIDs and so if the old 16-bit
storage should also be used for backward compatibility.)

  s.st_uid is 4 bytes

  s.st_gid is 4 bytes

-- UID not 2 bytes - disabling old 16-bit UID/GID support

Check for Large File Support

  off_t is 8 bytes

-- yes we have Large File Support!

Check for wide char support

-- have wchar_t - enabling Unicode support

Check for gcc no-builtin flag

Check for rmdir

Check for strchr

Check for strrchr

Check for rename

Check for mktemp

Check for mktime

Check for mkstemp

Check for memset

Check for memmove

Check for strerror

Check for errno declaration

Check for directory libraries

Check for readlink

Check for directory include file

Check for nonexistent include files

Check for term I/O include file

Check for valloc

Check for /usr/local/bin and /usr/local/man
```

```
Check for OS-specific flags
Check for symbolic links
```

The result of the **configure** script is the **Makefile** file. The **Makefile** file provides directions on how to build and install the software. The "**make**" section that follows provides details regarding the next steps in building the software package.

make

The **make** command uses a file named **Makefile** to perform specific operations. The **make** command is a utility for building and maintaining programs and other types of files from source code. A primary function of the **make** utility is to determine which pieces of a large program need to be recompiled and to issue the commands necessary to recompile them. Each **Makefile** is written by the software developer to perform operations related to the software project. Typically this includes the following types of functions:

- An operation to compile the software
- An operation to install the software
- An operation to "clean" previous versions of the compiled code

The following is an example of a simple **Makefile** that has only an install operation:

```
# more /usr/lib/firmware/Makefile
# This file implements the GNOME Build API:
# http://people.gnome.org/~walters/docs/build-api.txt

FIRMWAREDIR = /lib/firmware

all:

install:
        mkdir -p $(DESTDIR)$(FIRMWAREDIR)
        cp -r * $(DESTDIR)$(FIRMWAREDIR)
        rm -f $(DESTDIR)/usbdux/*dux $(DESTDIR)/*/*.asm
        rm $(DESTDIR)$(FIRMWAREDIR)/{WHENCE,LICENSE.*,LICENCE.*}
```

make install

The **install** option for the **make** command is designed to install code from source on the system. It may also include a compile process, depending on how the software developer created the **Makefile**. See the preceding "**make**" section for more details.

Storage Concepts

This section covers some of the essential concepts related to storage devices.

File Storage

With file storage, a remote storage device is used by the local system. The remote storage device is shared to the local system via a network filesystem structure, and users can place files and directories on the remote system. Common network filesystems include the following:

- **nfs:** This network-based filesystem originated on Unix systems. While it is an older filesystem, it has provided a standard way of sharing directory structures between Unix and Linux systems. Newer versions of this filesystem include modern security features and performance improvements.
- **smb:** This filesystem, which is also known as the Samba filesystem, is based on cifs and designed to provide network-based sharing.
- **cifs:** This filesystem is used on Microsoft Windows systems to share folders across the network. Samba utilities on Linux are used to connect to cifs shares.

Block Storage

A block storage device is a physical storage device and typically provides back-end storage for Linux storage systems. Examples of block storage devices include the following:

- Traditional SATA drives
- SSDs
- RAID drives

- Storage area networks (SANs), which can include Fibre Channel Protocol (FCP), Internet Small Computer System Interface (iSCSI), ATA over Ethernet (AoE), and Fibre Channel over Ethernet (FCoE)
- Optical discs (CD-ROM and DVD devices)

Data on block storage devices is accessible via a filesystem. Linux has a variety of local filesystem types, including the following:

- **ext3:** This filesystem, which is an extension of the ext2 filesystem, is designed to be placed on disk-based devices (partitions). While there are several differences between ext2 and ext3, the big change in ext3 was the introduction of journaling. Journaling helps prevent filesystem corruption by creating a log (journal) of changes made to files. In the event of a system crash, the recovery time of an ext3 filesystem should be relatively quick, as the journal can be used to quickly fix corrupted file metadata.
- **ext4:** The ext4 filesystem is a replacement for the ext3 filesystem. It supports larger filesystems and individual file sizes. Performance was improved in this version as well.
- **xfs:** This is another disk-based filesystem that is known for high performance and for handling larger file sizes.

Object Storage

Object storage is actually not a Linux concept; rather, it is typically found in cloud infrastructures. However, the object storage in a cloud infrastructure can be used by Linux, and this storage type is becoming a very popular way to store and share files.

Object storage is a feature in which objects (unstructured data like emails, videos, graphics, text, or any other type of data) can be stored in a cloud environment. Object storage doesn't use traditional filesystem storage features but rather organizes the data into "groups" (similar to a folder in a filesystem). Data is typically accessed using a URL much like one that you would use to access a web page. Object storage is durable and highly available, supports encryption, and can be used in a flexible manner that supports different backup and archiving features. Examples include AWS S3 (Simple Storage Service), Google Cloud Storage, and IBM Cloud Object Storage.

Partition Type

Partitions are used to separate a hard disk into smaller components. Each component can be treated as a different storage device, and a separate filesystem (btrfs, xfs, etx4, and so on) can be created on each partition.

There is a limit to the number of traditional PC-based partitions you can create. Originally only four partitions, referred to as *primary partitions*, were permitted. As more partitions were needed, a technique was created that makes it possible to convert one of the primary partitions into an extended partition. Within an extended partition, you can create additional partitions called *logical partitions*.

In Figure 1.5, **/dev/sda1**, **/dev/sda2**, and **/dev/sda3** are primary partitions. The **/dev/sda4** partition is an extended partition that is used as a container for the **/dev/sda5**, **/dev/sda6**, and **/dev/sda7** logical partitions.

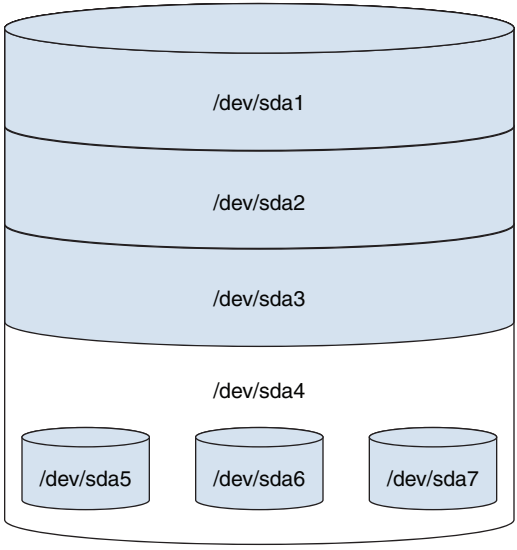

FIGURE 1.5 **Traditional Partition Structure**

On most Linux distributions that use traditional partitions, you are limited to a total of 15 partitions (though a kernel tweak can increase this number to 63).

Traditional partition tables are stored on the master boot record (MBR). A newer partition table, called the GUID partition table (GPT), does not face the same limitations or have the same layout as an MBR partition table.

Several different tools can be used to create or view partitions, including **fdisk**, **parted**, and the GUI-based tool provided by the installation program (which can vary based on the distribution).

Both **fdisk** and **parted** support command-line options, and both of them can be executed as interactive tools. These tools are covered in greater detail in Chapter 3, "Configure and Manage Storage Using the Appropriate Tools."

A *raw device* is a device file that is associated with a block device file (partition, hard disk, and so on). When you create this association, direct access to the block device is available. To create a raw device, use the **raw** command:

```
# raw /dev/raw/raw1 /dev/vda
/dev/raw/raw1: bound to major 252, minor 0
```

Once a raw device is created, you can use commands like the **dd** command to perform actions on the corresponding block file. The **dd** command is often used to create a copy of an entire hard disk.

Master Boot Record (MBR)

MBR partition tables are often referred to as "traditional" partitions, as opposed to newer partition tables such as the GUID partition table. An MBR partition table has the restriction of only permitting four partitions by default. This is an extremely limiting factor for operating systems such as Linux.

However, one of the primary partitions in an MBR partition table can be converted into an extended partition. Additional partitions, called *logical partitions*, can be added within this extended partition. See Figure 1.6 for a visual example.

A note regarding hard disk device names: Hard disks are referred to via device names in the **/dev** directory. IDE-based devices have names that start with **/dev/hd**, whereas SATA, SCSI, and USB devices have names that start with **/dev/sd**. The drives on a system are named starting from **a**, so the first SATA device would be **/dev/sda**, the second SATA device would be **/dev/sdb**, and so on. Partitions are numbered sequentially, starting from **1**, such as **/dev/sda1**, **/dev/sda2**, and **/dev/sda3**.

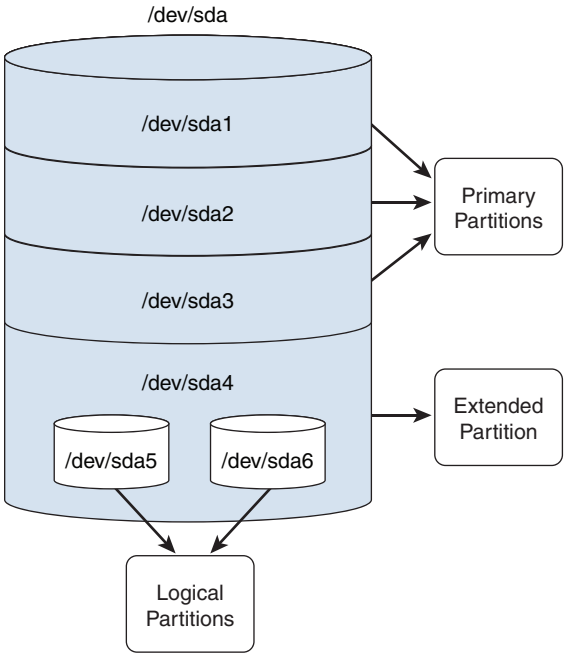

FIGURE 1.6 **Partitions**

GUID (Globally Unique Identifier) Partition Table (GPT)

The GUID partition table (GTP) is a partitioning scheme that is designed to overcome the limitations of MBR (see the "Master Boot Record (MBR)" section, earlier in this chapter). Unlike the MBR, the GPT doesn't have the limitation of four primary partitions. There also isn't a need for extended or logical partitions. The GPT supports up to 128 partitions per hard disk device.

Filesystem in Userspace (FUSE)

By default, only the root user can create filesystems. Filesystem in Userspace (FUSE) is a feature of the Linux kernel that allows regular users to create Linux filesystems.

ExamAlert

Setting up, configuring, and using FUSE is beyond the scope of the Linux+ XK0-005 exam.

Redundant Array of Independent (or Inexpensive) Disks (RAID) Levels

A RAID device is used to provide redundancy. Two or more physical devices can be combined to create a single device that stores data in a way that mitigates data loss in the event that one of the physical storage devices fails.

Note that RAID is covered in more detail in Chapter 3.

Striping

Striping, or RAID 0, is the process of writing to multiple drives as if they were a single device. The writes are performed using striping, in which some data is written to the first drive, then some data is written to the second drive, and so on. See Chapter 3 for more details.

Mirroring

With mirroring, or RAID 1, two or more disk drives appear to be one single storage device. Data that is written to one disk drive is also written to all of the other drives. If one drive fails, the data is still available on the other drives. See Chapter 3 for more details.

Parity

Parity data is a value that is derived from the data stored on the other devices in the RAID. Suppose there are three devices in the RAID, two of which (called devices A and B in this example) store regular filesystem data and one of which (called device C) stores the parity data. If device A fails, then the data that it stored could be rebuilt using a comparison of the data in device B and the parity data in device C. See Chapter 3 for more details.

Listing Hardware Information

This section focuses on several different utilities designed to display information about hardware devices or configure these devices.

lspci

The **lspci** command displays devices attached to the PCI bus.

Syntax:

lspci [*options*]

Here are some of the key options for the **lspci** command:

- **-b** is "bus centric," meaning it displays IRQ (Interrupt Request Line) numbers.
- **-n** displays device numbers rather than names; names typically are stored in **/usr/share/hwdata/pci.ids** or **/usr/share/hwdata/pci.ids.gz**.
- **-nn** displays both device numbers and names.
- **-v** shows verbose messages.
- **-vv** shows even more verbose messages.
- **-vvv** shows the most verbose messages.

Example:

```
[root@OCS ~]# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX
 PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA
 [Natoma/ Triton II]
00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4
 IDE (rev 01)
00:02.0 VGA compatible controller: InnoTek Systemberatung
 GmbH VirtualBox Graphics Adapter
00:03.0 Ethernet controller: Intel Corporation 82540EM
 Gigabit Ethernet Controller (rev 02)
00:04.0 System peripheral: InnoTek Systemberatung GmbH
 VirtualBox Guest Service
```

```
00:05.0 Multimedia audio controller: Intel Corporation
 82801AA AC'97 Audio Controller (rev 01)
00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB
00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR
 /FW/FRW (ICH6 Family) USB2 EHCI Controller
00:0d.0 SATA controller: Intel Corporation 82801HM/HEM
 (ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)
```

ExamAlert

The Linux+ XK0-005 exam should not ask specific questions on the output of the **lspci** command, but be ready to answer questions regarding the purpose of the command and the options described here.

lsusb

The **lsusb** command displays devices that are attached to the PCI bus.

Syntax:

lsusb [*options*]

Here are some of the key options for the **lsusb** command:

- **-D** displays a specific USB device (specified as an argument) rather than probing the **/dev/bus/usb** directory and displaying all USB devices.
- **-t** displays USB devices in a tree-like format.
- **-v** shows verbose messages.

Example:

```
[root@OCS Desktop]# lsusb
Bus 001 Device 002: ID 1221:3234 Unknown manufacturer Disk(Thumb
drive)
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

dmidecode

The **dmidecode** command is used to display a description of hardware components. In the following example, the **head** command was used to reduce the large amount of output:

```
[root@OCS ~]# dmidecode | head
# dmidecode 3.1
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 450 bytes.
Table at 0x000E1000.

Handle 0x0000, DMI type 0, 20 bytes
BIOS Information
        Vendor: innotek GmbH
        Version: VirtualBox
```

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. According to the FHS, where is information related to kernel data and process data stored?
   - ❍ **A. /tmp**
   - ❍ **B. /var**
   - ❍ **C. /usr/lib**
   - ❍ **D. /proc**

2. What is the third stage of the boot process?
   - ❍ **A.** BIOS
   - ❍ **B.** Kernel
   - ❍ **C.** Bootloader
   - ❍ **D.** System initialization

3. PXE uses a _____ server to obtain network configuration information, such as an IP address and subnet address.
   - ❍ **A.** DNS
   - ❍ **B.** NTP

- ❍ **C.** DHCP
- ❍ **D.** SAMBA

4. The **/dev/zero** file is a special file in Linux that returns _____ characters.
   - ❍ **A.** binary
   - ❍ **B.** zero-sized
   - ❍ **C.** null
   - ❍ **D.** None of these answers are correct.

Cram Quiz Answers

1. **D.** The **/proc** directory stores information related to kernel data and process data. The **/tmp** directory is the location for temporary files. The **/var** directory stores data that is variable in size. The **/usr/lib** directory stores libraries for the executables.
2. **B.** The kernel stage is the third stage. BIOS is the first, bootloader is the second, and system initialization is the fourth.
3. **C.** PXE uses a Dynamic Host Configuration Protocol (DHCP) server to obtain network configuration information, such as an IP address and subnet address. The other answers do not provide this configuration information.
4. **C.** The **/dev/zero** file is a special file in Linux that returns null characters.

This page intentionally left blank

CHAPTER 2

Manage Files and Directories

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.2:** Given a scenario, manage files and directories.

The focus of this chapter is on tools and concepts related to managing files and directories. The first section explores a variety of text editors. Many configuration files in Linux are in plaintext format, and knowing how to use text editors is critical for a Linux system administrator.

This chapter also covers how to archive and compress files, as well as how to copy files between different computer systems. This chapter explores a collection of file and directory commands that allow you to perform tasks such as moving, copying, creating, deleting, and displaying information about files and directories.

This chapter provides information on the following topics: file editing; file compression, archiving, and backup; file metadata; soft and hard links; copying files between systems; and file and directory operations.

File Editing

Most configuration files on Linux systems are in plaintext format. This makes it critical to know how to edit text files. This section focuses on common Linux editors: **sed**, **awk**, **printf**, **nano**, and the vi editor.

sed

Use the **sed** utility to make automated modifications to files. The basic format for the **sed** command is **sed** *'s/RE/string/' file*, where *RE* refers to the term *regular expression*, a feature that uses special characters to match patterns.

Here is an example of the **sed** command:

```
[student@OCS ~]$ head -n 5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
[student@OCS ~]$ head -n 5 /etc/passwd | sed 's/bin/----/'
root:x:0:0:root:/root:/----/bash
----:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/s----:/sbin/nologin
adm:x:3:4:adm:/var/adm:/s----/nologin
lp:x:4:7:lp:/var/spool/lpd:/s----/nologin
```

sed is a very powerful utility with a large number of features. Table 2.1 describes some of the most useful **sed** functions.

TABLE 2.1 **sed Functions**

Feature	Description
'/RE/d'	Deletes lines that match the RE from the output of the **sed** command.
'/RE/c*string***'**	Changes lines that match the RE to the value of *string*.
'/RE/a*string***'**	Adds *string* on a line after all lines that match the RE.
'/RE/i*string***'**	Adds *string* on a line before all lines that match the RE.

The **sed** command has two important modifiers (that is, characters added to the end of the **sed** operation):

- **g:** Means "global." By default, only the first RE pattern match is replaced. When the **g** modifier is used, all replacements are made. Figure 2.1 shows an example.
- **i:** Means "case-insensitive." This modifier matches an alpha character regardless of its case. So, the command **sed 's/a/-/i'** would match either **a** or **A** and replace it with the **-** character.

The **sed** command can also change the original file (instead of displaying the modified data to the screen). To change the original file, use the **-i** option.

```
[student@localhost ~]$ head -n 5 /etc/passwd | sed 's/bin/----/'
root:x:0:0:root:/root:/----/bash
----:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/s----:/sbin/nologin
adm:x:3:4:adm:/var/adm:/s----/nologin
lp:x:4:7:lp:/var/spool/lpd:/s----/nologin
[student@localhost ~]$ head -n 5 /etc/passwd | sed 's/bin/----/g'
root:x:0:0:root:/root:/----/bash
----:x:1:1:----:/----:/s----/nologin
daemon:x:2:2:daemon:/s----:/s----/nologin
adm:x:3:4:adm:/var/adm:/s----/nologin
lp:x:4:7:lp:/var/spool/lpd:/s----/nologin
```

FIGURE 2.1 **The g Modifier**

ExamAlert

The sed command is an extremely powerful tool with many features. For the Linux+ XK0-005 exam, focus on the features described in this section.

awk

The **awk** command is used to modify text that is in a simple database format. Consider the following example:

```
[student@OCS ~]$ head /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
```

With the **awk** command, you can break apart the data shown in this example into different fields and then perform actions on that data, as shown here:

```
[student@OCS ~]$ head /etc/passwd | awk -F: '{print $1,$7}'
root /bin/bash
bin /sbin/nologin
daemon /sbin/nologin
adm /sbin/nologin
lp /sbin/nologin
sync /bin/sync
shutdown /sbin/shutdown
halt /sbin/halt
mail /sbin/nologin
operator /sbin/nologin
```

In this example, the **-F** option is used as a field separator, and each field is assigned to a variable, as described in Table 2.2.

TABLE 2.2 **awk Field Variables**

Variable	Description
$1, **$2**, and so on	The field variables
$0	The entire line
NF	The number of fields on the line (Do not use the **$** character for this variable.)
NR	The current line number

Table 2.3 lists some important options of the **awk** command.

TABLE 2.3 **Important awk Command Options**

Option	Description
-F	Used to specify the field separator.
-f	Used to specify a file that contains the **awk** commands to execute.

printf

The **printf** command is sometimes used by BASH scripters to format data before displaying it to the user who is running the script. This command can help you format data such as digits and strings. For example, compare the

output of the following two commands (where the **echo** command is a simpler text-printing command):

```
[student@OCS ~]$ total=10.000
[student@OCS ~]$ echo "$total"
10.000
[student@OCS ~]$ printf "%5f \n" $total
10.000000
```

Note how **%5f** changed the precision of the number to be five places after the decimal point. Also note that **\n** produced a newline character, which, unlike with the **echo** command, is not automatically printed.

nano

The **nano** editor is a non-GUI editor that provides a handy "cheat sheet" at the bottom of the screen. See Figure 2.2 for an example.

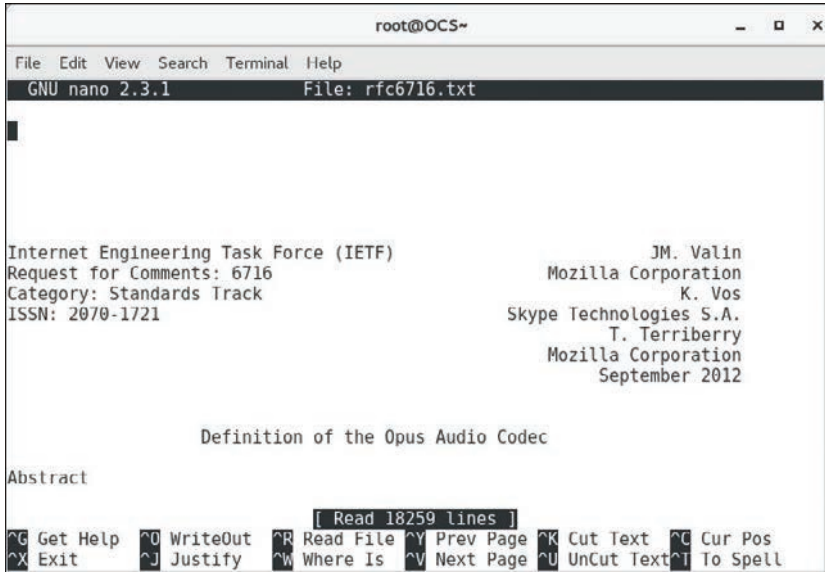

FIGURE 2.2 The nano Editor

You can find commands in **nano** by holding down the Ctrl key and then pressing another key. For example, **Ctrl-x** exits the **nano** editor. In the list of commands at the bottom of the screen, the **^** symbol represents the Ctrl key. Also note that **^X** doesn't mean **Ctrl-Shift-x**; rather, it means just **Ctrl-x**.

You should be aware of the commonly used **nano** commands in Table 2.4.

TABLE 2.4 **Commonly Used nano Commands**

Command	Description
^G	Displays a help document. Note that when the help document appears, different commands are displayed at the bottom of the screen.
^O	Saves a file.
^X	Exits the editor.
^K	Cuts a line (that is, deletes the line and places it in memory). **^K** can be used to cut multiple lines into memory.
^U	Uncuts a line. Note that this isn't an undo operation but rather a paste operation.
^^	Marks the beginning and ending text. Marked text can be copied using **Alt-^**. Use one **^^** to start the marking, use arrow keys to select the text to mark, and then use a final **^^** to end the marking. Note that in **^^**, the second **^** is an actual **^** character, and the first **^** represents the control character.
^W	Searches for text in the current document.
^F	Moves forward one screen.
^B	Moves back one screen.
^n	Moves to a specific line number. For example, **^5** moves to line 5.
^C	Displays the current position in the document.

vi

The vi editor is a standard text editor for both Linux and Unix environments. Although it may not be as user friendly as other editors, it has a few important advantages:

- The vi editor (or vim, which is an improved version of the vi editor) is on every Linux distribution (and all Unix flavors). This means if you know how to edit files with the vi editor, you can always edit a file, regardless of which distribution you are working on.
- Because the vi editor is a command-line-only editor, it does not require a graphical user interface (GUI). This is important because many Linux servers do not have a GUI installed, which means you can't use GUI-based text editors.
- When you understand vi well, you will find that it is an efficient editor, and you can edit files very quickly using it compared to using most

other editors. All vi commands are short and keyboard based, and you don't have to spend time taking your hands off the keyboard to use the mouse.

To edit a new file with the vi editor, you can just type the command with no arguments or type **vi** *filename*.

> **Note**
>
> The vim editor is an improved text editor that has additional features that are not available in the vi editor. Many Linux distributions use the vim editor by default. One advantage of the vim editor is that it includes all the features and commands of the vi editor. So, if you learned to use the vi editor 30 years ago, your knowledge will still apply in the vim editor. All of the commands in this chapter work in both the vi and vim editors.

The vi editor was designed to only use a keyboard, which poses a challenge because sometimes a key should execute a command and sometimes a key should represent a character to insert into the document. To allow the keys to perform different tasks, vi has three modes of operation:

- **Command mode:** This is the default mode, and when you open vi, you are placed in the command mode. In this mode, you can perform commands that can move around the screen, delete text, and paste text.
- **Insert mode:** While you're in insert mode, any key typed appears in your document as new text. When you are finished adding new text, you can return to the default mode (that is, the command mode) by pressing the Esc key.
- **Last line mode:** This mode, also called the *ex mode*, allows you to perform more complex operations, such as saving a document to a file with a different name. To enter last line mode from the command mode, press the **:** key. After you enter a command and press Enter, the command is executed, and normally you are returned to the command mode. In some cases, you may need to press the Esc key to return to the command mode.

> **Note**
>
> You cannot move from the insert mode to the last line mode or vice versa. To move to the insert mode or the last line mode, you first must be in the command mode. Pressing the Esc key places you in the command mode.

To search for files while in the vi editor, you can use either the **/** or the **?** character when you are working in the command mode. When you type either the **/** or the **?** character, a prompt appears in the bottom-left portion of the screen, as shown in Figure 2.3.

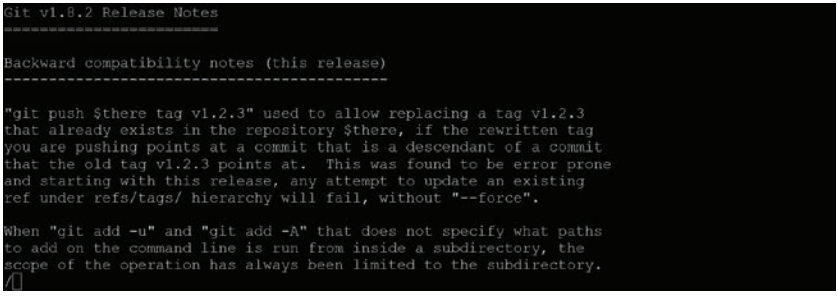

FIGURE 2.3 **Searching for Files in the vi Editor**

At the **/** or **?** prompt, type the value you want to search for. You can use regular expressions within this value. For example, if you type **/^the**, the vi editor searches for the next occurrence of the string **"the"** that appears at the beginning of a line.

The **/** character performs a forward search, and **?** searches backward in the document. If you don't find what you are searching for, you can use the **n** character to find the next match. When the search is started with a **/** character, the **n** character continues the search forward in the document. When the search is started with a **?** character, the **n** character continues the search backward in the document.

To reverse the current search, use the **N** character. For example, suppose you start the search by typing **?^the**. Using the **N** character results in searching forward in the document.

While in the command mode, you can move around the screen by using the keys described in Table 2.5.

TABLE 2.5 **vi Command Mode Navigation Keys**

Key	Description
h	Moves one character to the left.
j	Moves one line down.
k	Moves one line up.
l	Moves one character to the right.

While in the command mode, you can enter the insert mode by using one of the keys described in Table 2.6.

TABLE 2.6 **vi Insert Mode Navigation Keys**

Key	Description
i	Enters the insert mode before the character the cursor is on.
I	Enters the insert mode before the beginning of the current line.
o	Opens a new line below the current line and enters the insert mode.
O	Opens a line above the current line.
a	Enters the insert mode after the character the cursor is on.
A	Enters the insert mode after the end of the current line.

While in the command mode, you can modify text by using the keys described in Table 2.7.

TABLE 2.7 **vi Command Mode Keys to Modify Text**

Key	Description
c	The **c** key is combined with other keys to change data. For example, **cw** changes the current word, and **c$** changes from the cursor position to the end of the line. When finished making your changes, press the Esc key.
d	The **d** key is combined with other keys to delete data. For example, **dw** deletes the current word, and **d$** deletes from the cursor position to the end of the line. All of the deleted data is stored in a buffer and can be pasted back into the document with the **p** or **P** key.
p	After cutting text with the **d** key or copying text with the **y** key, you can paste with the **p** or **P** key. A lowercase **p** pastes after the current cursor, whereas an uppercase **P** pastes before the cursor.
y	The **y** key is combined with other keys to copy data. For example, **yw** copies the current word, and **y$** copies from the cursor position to the end of the line. All of the copied data is stored in a buffer and can be pasted back into the document with the **p** or **P** key.
dd	The **dd** command deletes the current line.
yy	The **yy** command copies the current line.

While in the command mode, you can save and/or quit a document by using the keys described in Table 2.8.

TABLE 2.8 **vi Command Mode Keys to Save/Quit a Document**

Key	Description
ZZ	Saves and quits the document. This is equivalent to **:wq**.
:w!	Forces the vi editor to write changes in the document to the file.

Key	Description
:q!	Forces the vi editor to quit, even if changes in the file have not been saved.
:e!	Opens a new file to edit and forgets all changes in the document since the last write. This requires a filename argument as an option (for example, **:e! myfile.txt**).

ExamAlert

Expect the Linux+ XK0-005 exam to include more questions about the vi editor than about the nano editor.

File Compression, Archiving, and Backup

When disaster strikes, such as a hard disk failure or a natural disaster, data may become corrupted. Using archive and restore utilities helps limit the risks involved with data loss and makes it easier to transfer information from one system to another. This section focuses on these utilities.

ExamAlert

Many of the options for compression commands **(gzip, bzip2**, and **zip)** are the same. This makes them easier to remember for the Linux+ XK0-005 exam.

gzip

Use the **gzip** command to compress files, as shown here:

```
[student@OCS ~]$ ls -lh juju
-rwxr-xr-x 1 vagrant vagrant 109M Jan 10 09:20 juju
[student@OCS ~]$ gzip juju
[student@OCS ~]$ ls -lh juju.gz
-rwxr-xr-x 1 vagrant vagrant 17M Jan 10 09:20 juju.gz
```

Note that the **gzip** command replaces the original file with the compressed file.

Table 2.9 details some important **gzip** options.

TABLE 2.9 **gzip Command Options**

Option	Description
-c	Writes output to stdout and does not replace the original file. You can use redirection to place output data into a new file (for example, **gzip -c juju > juju.gz**).
-d	Decompresses the file. (You can also use the **gunzip** command for decompression.)
-r	Compresses recursively (that is, compresses all files in the directory and its subdirectories).
-v	Enters verbose mode and displays a percentage of compression.

ExamAlert

The **gzip**, **xz**, and **bzip2** commands are very similar to one another. The biggest difference is the technique used to compress files. The **gzip** command uses the Lempel-Ziv coding method, whereas the **bzip2** command uses the Burrows-Wheeler block-sorting text-compression algorithm and Huffman coding. The **xz** command uses the LZMA and LZMA2 compression methods.

Use the **gunzip** command to decompress gzipped files, as shown here:

```
[student@OCS ~]$ ls -lh juju.gz
-rwxr-xr-x 1 vagrant vagrant 17M Jan 10 09:20 juju.gz
[student@OCS ~]$ gunzip juju
[student@OCS ~]$ ls -lh juju
-rwxr-xr-x 1 vagrant vagrant 109M Jan 10 09:20 juju
```

bzip2

Use the **bzip2** command to compress files, as shown here:

```
[student@OCS ~]$ ls -lh juju
-rwxr-xr-x 1 vagrant vagrant 109M Jan 10 09:20 juju
[student@OCS ~]$ bzip2 juju
[student@OCS ~]$ ls -lh juju.bz2
-rwxr-xr-x 1 vagrant vagrant 14M Jan 10 09:20 juju.bz2
```

Note that the **bzip2** command replaces the original file with the compressed file.

Table 2.10 details some important **bzip2** options.

TABLE 2.10 **bzip2 Command Options**

Option	Description
-c	Writes output to stdout and does not replace the original file. You can use redirection to place output data into a new file (for example, **bzip2 -c juju > juju.bz**).
-d	Decompresses the file. (You can also use the **bunzip2** command.)
-v	Enters verbose mode and displays a percentage of compression.

ExamAlert

The **gzip**, **xz**, and **bzip2** commands are very similar to one another. The biggest difference is the technique used to compress files. The **gzip** command uses the Lempel-Ziv coding method, whereas the **bzip2** command uses the Burrows-Wheeler block-sorting text-compression algorithm and Huffman coding. The **xz** command uses the LZMA and LZMA2 compression methods.

zip

The **zip** command is used to merge multiple files into a single compressed file. To create a compressed file named **mail.zip** that contains all the files in the **/etc/mail** directory, use the following format:

```
[student@OCS ~]$ zip mail /etc/mail*
 adding: etc/mail/ (stored 0%)
 adding: etc/mailcap (deflated 53%)
 adding: etc/mailman/ (stored 0%)
 adding: etc/mail.rc (deflated 49%)
```

Table 2.11 details some important **zip** options.

TABLE 2.11 **zip Command Options**

Option	Description
-d	Decompresses the file. (You can also use the **unzip** command.) Note that the zipped file is not deleted.
-v	Enters verbose mode and displays percentage of compression.
-u	Updates a **.zip** file with new content.
-r	Zips recursively, meaning you can specify a directory, and all of the contents in that directory (including all subdirectories and their contents) are zipped.
-x *file(s)*	Excludes the specified *file(s)* from the **.zip** file.

ExamAlert

Remember that **zip** merges multiple files together, whereas **bzip2** and **gzip** do not.

tar

The purpose of the **tar** command is to merge multiple files into a single file. To create a tar file named **sample.tar**, execute the following:

```
tar -cf sample.tar files_to_merge
```

To list the contents of a **.tar** file, execute the following:

```
tar -tf sample.tar
```

To extract the contents of a **.tar** file, execute the following:

```
tar -xf sample.tar
```

Table 2.12 details some important **tar** options.

TABLE 2.12 **tar Command Options**

Option	Description
-c	Creates a **.tar** file.
-t	Lists the contents of a **.tar** file.
-x	Extracts the contents of a **.tar** file.
-f	Specifies the name of the **.tar** file.
-v	Enters verbose mode and provides more details about what the command is doing.
-A	Appends new files to an existing **.tar** file.
-d	Compares a **.tar** file and the files in a directory and determines the differences between them.
-u	Updates by appending newer files to an existing **.tar** file.
-j	Compresses/uncompresses the **.tar** file using the **bzip2** utility.
-J	Compresses/uncompresses the **.tar** file using the **xz** utility.
-z	Compresses/uncompresses the **.tar** file using the **gzip** utility.

xz

Use the **xz** command to compress files, as shown here:

```
[student@OCS ~]$ ls -lh juju
-rwxr-xr-x 1 vagrant vagrant 109M Jan 10 09:20 juju
[student@OCS ~]$ xz juju
[student@OCS ~]$ ls -lh juju.xz
-rwxr-xr-x 1 vagrant vagrant 11M Jan 10 09:20 juju.xz
```

Table 2.13 details some important **xz** options.

TABLE 2.13 **xz Command Options**

Option	Description
-c	Writes output to stdout and does not replace the original file. You can use redirection to place output data into a new file (for example, **xz -c juju > juju.xz**).
-d	Decompresses the file. (You can also use the **unxz** command.)
-l	Lists information about an existing compressed file (for example, **xz -l juju.xz**).
-v	Enters verbose mode and displays the percentage of compression.

ExamAlert

The **gzip**, **xz**, and **bzip2** commands are very similar to one another. One noticeable difference is the technique used to compress files. The **gzip** command uses the Lempel-Ziv coding method, whereas the bzip2 command uses the Burrows-Wheeler block-sorting text-compression algorithm and Huffman coding. The **xz** command uses the LZMA and LZMA2 compression methods.

cpio

The purpose of the **cpio** command is to create archives. You can create an archive of files by sending the filenames into the command as stdin, as in the following example:

```
[student@OCS ~]$ find /etc -name "*.conf" | cpio -ov > conf.cpio
```

Table 2.14 details some important **cpio** options.

TABLE 2.14 **cpio Command Options**

Option	Description
-d	Used with the **-i** option to extract the directory structure as well as the files in the **cpio** file.
-i	Extracts data from a **cpio** file; the file should be provided via STDIN (for example, **cpio -i < conf.cpio**).
-o	Creates an archive (output file).
-t	Lists the table of contents of a **cpio** file.
-v	Enters verbose mode.

dd

The **dd** command can perform multiple operations related to backing up data and creating files. One common use is to make a backup of an entire drive; for example, the following command backs up the entire **/dev/sdb** device to the **/dev/sdc** device:

```
[student@OCS ~]$ dd if=/dev/sdb of=/dev/sdc bs=4096
```

Another use of the **dd** command is to create a large file that can be used as a swap file:

```
[student@OCS ~]$ dd if=/dev/zero of=/var/swapfile bs=1M count=50
```

Table 2.15 details some important **dd** options.

TABLE 2.15 **dd Command Options**

Option	Description
if=	Specifies the input file.
of=	Specifies the output file.
bs=	Specifies the block size.
count=	Indicates the number of blocks to create/transfer.

File Metadata

File metadata is information about a file, other than the file contents. Two useful commands for displaying file metadata are covered in this section: the **stat** command and the **file** command.

stat

A file or directory consists of several components. Many of these components, such as the owner and permissions, are stored in a filesystem element called an inode.

Everything about a file except for the data in the file and the filename is stored in the inode. Each file is given an inode number that is unique for the filesystem in which the file resides.

The inode of a file contains the following information:

- Unique inode number
- User owner
- Group owner
- Mode (permissions and file type)
- File size
- Timestamps:
  - Last time the file contents were modified
  - Last time the inode data was modified
  - Last time the file was accessed
- Pointers (references to the data block locations that contain the file data)

You can see this inode information with the **stat** command, as shown here:

```
[root@OCS ~]$ stat /etc/passwd
 File: '/etc/passwd'
 Size: 2597 Blocks: 8 IO Block: 4096 regular file
Device: fc01h/64513d Inode: 33857 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid:
 ( 0/ root)
Access: 2018-10-12 12:54:01.126401594 -0700
Modify: 2018-09-08 12:53:48.371710687 -0700
Change: 2018-09-08 12:53:48.371710687 -0700
 Birth: -
```

file

The **file** command reports the type of contents in a file. Here are some examples:

```
[student@localhost ~]$ file /etc/hosts
/etc/hosts: ASCII text
[student@localhost ~]$ file /usr/bin/ls
/usr/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked (uses shared libs), for GNU/Linux 2.6.32,
BuildID[sha1]=aa7ff68f13de25936a098016243ce57c3c982e06, stripped
[student@localhost ~]$ file /usr/share/doc/pam-1.1.8/html/
sag-author.html
/usr/share/doc/pam-1.1.8/html/sag-author.html: HTML document,
UTF-8 Unicode text, with very long lines
```

Soft and Hard Links

There are two different types of link files: hard links and soft (also called symbolic) links. Understanding these link types is important when determining if you should link a file or make a file copy. This section covers the purposes of links and how to create links using the **ln** command.

Symbolic (Soft) Links

When you create a soft link, the original file contains the data, and the link file points to the original file. Any changes made to the original file also appear to be in the linked file because using the linked file always results in following the link to the target file. Deleting the original file results in a broken link, making the link file worthless and resulting in complete data loss.

Figure 2.4 demonstrates soft links.

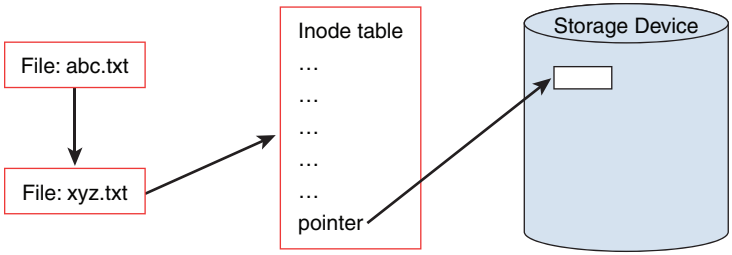

FIGURE 2.4 **Soft Links**

In Figure 2.4, the **abc.txt** file is soft-linked to the **xyz.txt** file. The **abc.txt** file points to the filename **xyz.txt**, not the same inode table. (Although not shown in this figure, the **abc.txt** file has its own inode table.) When the process that is accessing the link file follows the link, the data for the **xyz.txt** file is accessible via the **abc.txt** file.

Copying files results in a complete and separate copy of the data. Changes in the original file have no effect on the data in the copied file. Changes in the copied file have no effect on the data in the original file. Deleting one of these files has no impact on the other file.

To create a link, execute the **ln** command in the following manner: **ln [-s]** *target_file link_file*. You can create a soft link to any file or directory:

```
[root@OCS ~]$ ln -s /boot/initrd.img-3.16.0-30-generic initrd
```

The **ls** command can be used to view both soft and hard links. Soft links are very easy to see because the target file is displayed when executing the **ls -l** command, as shown here:

```
[root@OCS ~]$ ls -l /etc/vtrgb
lrwxrwxrwx 1 root root 23 Jul 11 2015
 /etc/vtrgb -> /etc/alternatives/vtrgb
```

Hard Links

When you create a hard link to a file, there is no way to distinguish the "original" file from the "linked" file. They are just two filenames that point to the same inode and, therefore, the same data. (See the "inodes" section in this chapter for more details.) If you have 10 hard-linked files and you delete any 9 of these files, the data is still maintained in the remaining file.

Figure 2.5 demonstrates hard links.

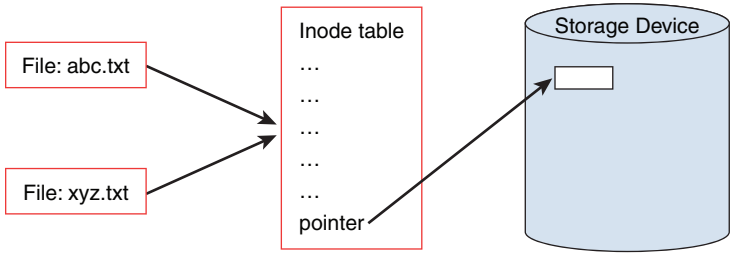

FIGURE 2.5 **Hard Links**

In Figure 2.5, the **abc.txt** and **xyz.txt** files are hard-linked together. This means that they share the same inode tables. An **…** in the inode table represents metadata—information about the file such as the user owner and permissions. Included with this metadata are pointers that refer to blocks within the storage device where the file data is stored.

To create a link, execute the **ln** command in the following manner: **ln [-s]** *target_file link_file*. For example, to create a hard link from the **/etc/hosts** file to a file in the current directory called **myhosts**, execute the following command:

```
[root@OCS ~]$ ln /etc/hosts myhosts
```

Hard-linked files share the same inode. You can only make a hard link to a file (not a directory) that resides on the same filesystem as the original file. Creating hard links to files on another filesystem or to directories results in errors, as shown here:

```
[root@OCS ~]$ ln /boot/initrd.img-3.16.0-30-generic initrd
ln: failed to create hard link 'initrd' =>
 '/boot/initrd.img-3.16. 0-30-generic': Invalid cross-device link
[root@OCS ~]$ ln /etc myetc
ln: '/etc': hard link not allowed for directory
```

The **ls** command can be used to view both soft and hard links. Hard links are more difficult because a hard link file shares an inode with another filename. For example, the value **2** after the permissions in the following output indicates that this is a hard link file:

```
[root@OCS ~]$ ls -l myhosts
-rw-r--r-- 2 root root 186 Jul 11 2015 myhosts
```

To view the inode number of a file, use the **-i** option to the **ls** command:

```
[root@OCS ~]$ ls -i myhosts
263402 myhosts
```

Then use the **find** command to search for files with the same inode:

```
[root@OCS ~]$ find / -inum 263402 -ls 2>/dev/null
263402 4 -rw-r--r-- 2 root root 186 Jul 11 2015 /root/myhosts
263402 4 -rw-r--r-- 2 root root 186 Jul 11 2015 /etc/hosts
```

ExamAlert

Know the difference between hard and soft links. You are likely to get a question on the Linux+ XK0-005 exam that tests your understanding of the differences.

Copying Files Between Systems

Any filesystem is bound to have thousands of files and directories that need to be managed. This section focuses on the Linux commands used to manage these filesystem objects.

rsync

The **rsync** command is useful in copying files remotely across the network. It is typically used in situations where changes from previous files need to be copied over because it handles this more efficiently than other remote copy methods.

Syntax:

```
rsync [options] source destination
```

Table 2.16 describes some important **rsync** options.

TABLE 2.16 **rsync Command Options**

Option	Description
-t	Preserves the original modification timestamp.
-v	Enters verbose mode.
-r	Copies recursively (that is, transfers entire directories).

Option	Description
-l	Maintains symbolic links.
-p	Preserves original permissions.

scp

The **scp** command is used to copy files to and from remote systems via the Secure Shell service. To copy a file from your local machine to a remote machine, use the following syntax:

```
scp filename user@machine:/directory
```

In this syntax, *user* is an account name on the remote system, *machine* is the remote system, and */directory* represents where you want to store the file.

Table 2.17 describes some important options of the **scp** command.

TABLE 2.17 **scp Command Options**

Option	Description
-P *port*	Specifies the port number to connect to. Typically SSH servers use port 22, and that is the default for the **scp** command.
-p	Attempts to preserve the timestamps and permissions of the original file.
-r	Recursively copies entire directories.
-v	Enters verbose mode.

nc

The man page of the **nc** command provides an excellent summary of the **nc** command:

> The nc (or netcat) utility is used for just about anything under the sun involving TCP or UDP. It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and IPv6. Unlike telnet(1), nc scripts nicely, and separates error messages onto standard error instead of sending them to standard output, as telnet(1) does with some.

There are quite a few uses for the **nc** command. For example, suppose you want to know if a specific port is being blocked by your company firewall before you bring online a service that makes use of this port. On the internal server, you can have the **nc** command listen for connections on that port:

```
[root@server ~]# nc -l 3333
```

The result should be a blank line below the **nc** command. Next, to connect, on a remote system outside your network, you could run the following **nc** command (replacing *server* with the resolvable hostname or IP address of the local system):

```
[root@client Desktop]# nc server 3333
```

If the connection is established, you see a blank line under the **nc** command line. If you type something on this blank line and press the Enter key, then what you typed appears below the **nc** command on the server. Actually, the communication works both ways: What you type on the server below the **nc** command appears on the client as well.

The following are some useful options to the **nc** command:

- **-w:** This option is used on the client side to close a connection automatically after a timeout value is reached. For example, **nc -w 30 server 333** closes the connection 30 seconds after it has been established.
- **-6:** Use this option to enable IPv6 connections.
- **-k:** Use this option to keep server processes active, even after the client disconnects. The default behavior is to stop the server process when the client disconnects.
- **-u:** Use UDP connections rather than TCP connections (the default). This is important for correctly testing firewall configurations as a UDP port might not be blocked while the TCP port is blocked.

You can also use the **nc** command to display open ports, much the way you use the **netstat** command:

```
[root@onecoursesource Desktop]# nc -z localhost 1000-4000
Connection to localhost 3260 port [tcp/iscsi-target] succeeded!
Connection to localhost 3333 port [tcp/dec-notes] succeeded!
```

The **-z** option can also be used to port scan a remote host.

There is a useful way to use the **nc** command to transfer all sorts of data. The format you use, assuming that the transfer is from the client to the server, is shown below (where you replace *cmd* with an actual command):

On the server: **nc -l 3333 |** *cmd*

On the client: *cmd* **| nc server 3333**

For example, you can transfer an entire **/home** directory structure from the client to the server with the **tar** command by first executing the following on the server:

```
nc -l 333 | tar xvf -
```

Then, on the client command, you execute the following command:

```
tar cvf - /home | nc server 333
```

The client merges the contents of the **/home** directory structure into a tar ball. The **-** tells the **tar** command to send this output to standard output. The data is sent to the server via the client's **nc** command, and then the server's **nc** command sends this data to the **tar** command. As a result, the **/home** directory from the client is copied into the current directory of the server.

File and Directory Operations

mv

The **mv** command moves or renames a file.

Example:

```
mv /tmp/myfile ~
```

Table 2.18 describes some important options of the **mv** command.

TABLE 2.18 **mv Command Options**

Option	Description
-i	Provides an interactive prompt if the move process would result in overwriting an existing file.
-n	Prevents an existing file from being overwritten.
-v	Enters verbose mode and describes actions taken when moving files and directories.

cp

The **cp** command is used to copy files or directories. Here's the syntax for this command:

```
cp [options] file|directory destination
```

file|*directory* is the file or directory to copy. *destination* is where to copy the file or directory to. The following example copies the **/etc/hosts** file into the current directory:

```
[student@OCS ~]$ cp /etc/hosts .
```

Note that the destination *must* be specified (hence the **.** character that represents the current directory in this example).

Table 2.19 describes some important options of the **cp** command.

TABLE 2.19 **cp Command Options**

Option	Description
-i	Provides an interactive prompt if the copy process results in overwriting an existing file.
-n	Prevents an existing file from being overwritten.
-r	Recursively copies the entire directory structure.
-v	Enters verbose mode and describes actions taken when copying files and directories.

mkdir

The **mkdir** command creates a directory.

Example:

```
mkdir test
```

Table 2.20 describes some important **mkdir** options.

TABLE 2.20 **mkdir Command Options**

Option	Description
-m *perm*	Sets the permissions for the new directory rather than using the umask value.
-p	Creates parent directories, if necessary; for example, **mkdir /home/bob/data/january** creates all the directories in the path specified if they don't already exist.
-v	Enters verbose mode and prints a message for every directory that is created.

rmdir

The **rmdir** command is used to delete empty directories. This command fails if the directory is not empty. (Use **rm -r** to delete a directory and all the files within the directory.)

Example:

```
rmdir data
```

ls

The **ls** command is used to list files in a directory. Table 2.21 describes some important options of the **ls** command.

TABLE 2.21 **ls Command Options**

Option	Description
-a	Lists all files, including hidden files.
-d	Lists the directory name, not the contents of the directory.
-F	Appends a character to the end of the file to indicate its type; examples include * for an executable file, / for a directory, and @ for a symbolic link file.
-h	When used with the **-l** option, provides file sizes in human-readable format.
-i	Displays each file's inode value.
-l	Displays a long listing (refer to the figure).
-r	Reverses the output order of the file listing.
-S	Sorts by file size.
-t	Sorts by modification time (with newest files listed first).

The output of the **ls -l** command includes one line per file, as demonstrated in Figure 2.6.

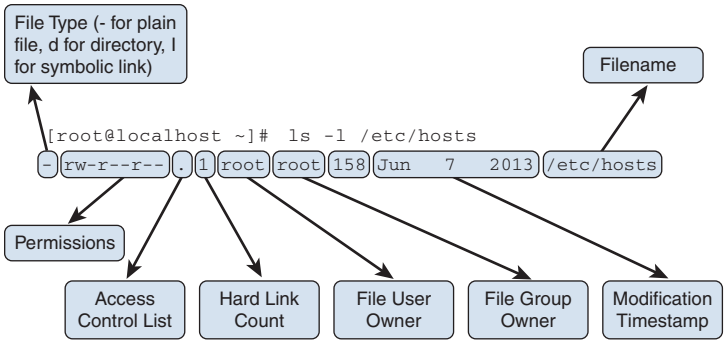

FIGURE 2.6 **The ls -l Command**

pwd

The **pwd** ("print working directory") command displays the shell's current directory, as in this example:

```
[student@localhost rc0.d]$ pwd
/etc/rc0.d
```

rm

The **rm** command is used to delete files and directories.

Example:

```
rm file.txt
```

Table 2.22 describes some important options of the **rm** command.

TABLE 2.22 **rm Command Options**

Option	Description
-i	Provides an interactive prompt before removing the file.
-r	Recursively deletes the entire directory structure.
-v	Enters verbose mode and describes actions taken when deleting files and directories.

cd

To move the shell's current directory to another directory, use the **cd** ("change directory") command. The **cd** command accepts a single argument: the

location of the desired directory. For example, to move to the **/etc** directory, you can execute the following command:

```
[student@localhost ~]$ cd /etc
[student@localhost etc]$
```

The **cd** command is "no news is good news" sort of command. If the command succeeds, no output is displayed (although the prompt changes). If the command fails, an error is displayed, as shown below:

```
[student@localhost ~]$ cd /etc
bash: cd: nodir: No such file or directory
[student@localhost ~]$
```

. (Current Directory)

One dot (period) represents the current directory. This isn't very useful with the **cd** command, but it is handy with other commands when you want to say "the directory I am currently in."

.. (Level Above the Current Directory)

Two dot characters represent one level above the current directory. So, if the current directory is **/etc/skel**, the command **cd ..** changes the current directory to the **/etc** directory.

~ (User's Home Directory)

The tilde character represents the user's home directory. Every user has a home directory (typically **/home/***username*) for storing their own files. The **cd ~** command returns you to your home directory.

tree

The **tree** command allows you to see a directory hierarchy, as in this example:

```
[student@localhost ~]$ tree /etc | head -20
/etc
|-- abrt
|   |-- abrt-action-save-package-data.conf
```

```
|   |-- abrt.conf
|   |-- abrt-harvest-vmcore.conf
|   |-- gpg_keys.conf
|   |-- plugins
|   |   |-- CCpp.conf
|   |   `-- python.conf
|   `-- xorg.conf
|-- acpi
|   |-- actions
|   |   `-- power.sh
|   `-- events
|       |-- powerconf
|       `-- videoconf
|-- adjtime
|-- aide.conf
|-- aliases
|-- aliases.db
```

cat

The **cat** command displays the contents of text files. Table 2.23 describes some important **cat** command options.

TABLE 2.23 **cat Command Options**

Option	Description
-A	Functions the same as **-vET**.
-e	Functions the same as **-vE**.
-E	Displays a $ character at the end of each line (to visualize trailing whitespace characters).
-n	Numbers all lines of output.
-s	Converts multiple blank lines into a single blank line.
-T	Displays ^I for each tab character (in order to show spaces instead of tabs).
-v	Displays “unprintable” characters (such as control characters).

ExamAlert

The **cat** command does not pause the display after one page of output.

touch

The **touch** command has two functions: to create an empty file and to update the modification and access timestamps of an existing file. To create a file or update an existing file's timestamps to the current time, use the following syntax:

```
touch filename
```

Table 2.24 describes some important options of the **touch** command.

TABLE 2.24 **touch Command Options**

Option	Description
-a	Modifies the access timestamp only, not the modification timestamp.
-d *DATE*	Sets the timestamp to the specified *DATE* (for example, **touch -d "2018-01-01 14:00:00"**).
-m	Modifies the modification timestamp only, not the access timestamp.
-r *file*	Uses the timestamp of *file* as a reference to set the timestamps of the specified file (for example, **touch -r /etc/hosts /etc/passwd**).

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. While of the following is a valid **sed** command?
   - ❍ **A. ls -l | sed 's~root~null~g'**
   - ❍ **B. ls -l | sed 's\root\null\g'**
   - ❍ **C. ls -l | sed 's-root-null-g'**
   - ❍ **D. ls -l | sed 's/root/null/g'**

2. Which of the following tools can be used to merge multiple files together? (Choose two.)
   - ❍ **A. zip**
   - ❍ **B. gzip**
   - ❍ **C. bzip**
   - ❍ **D. tar**

3. Which option to the **ln** command creates a hard link?
   - ❍ **A. -s**
   - ❍ **B. -h**
   - ❍ **C. -l**
   - ❍ **D.** None of these answers are correct.

4. Which option to the **ls** command displays all files, including hidden files?
   - ❍ **A. -l**
   - ❍ **B. -a**
   - ❍ **C. -d**
   - ❍ **D. -s**

Cram Quiz Answers

1. **D.** The **/** character should be used between the different values of a **sed** statement.
2. **A and D.** The **zip** and **tar** commands merge multiple files together. The **gzip** and **bzip2** commands compress files individually.
3. **D.** The **ln** command creates hard links without the need for any options. To create soft links, you must use the **-s** option.
4. **B.** Use the **-a** option to display all files, including hidden files.

CHAPTER 3

Configure and Manage Storage Using the Appropriate Tools

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.3:** Given a scenario, configure and manage storage using the appropriate tools.

In this chapter you will learn how to manage storage devices. The first section covers disk partitioning, and you will learn about tools like **fdisk, parted**, and **partprobe**. Once you have learned how to create a partition, you will learn about how to place a filesystem on the partition and make the filesystem available to the operating system through a process called *mounting*.

This chapter explores two alternatives to partitioning: Logical Volume Manager and RAID. This chapter also covers the essentials of working with remote storage devices, such as NFS (Network File System) and SMB/CIFS (Server Message Block/Common Internet File System).

This chapter provides information on the following topics: disk partitioning, mounting of local and remote devices, filesystem management, monitoring of storage space and disk usage, creation and modification of volumes using Logical Volume Manager (LVM), inspection of RAID implementations, storage area network (SAN)/network-attached storage (NAS), and storage hardware.

Disk Partitioning

Partitions are used to separate a hard disk into smaller components. Each component can be treated as a different storage device, and a separate filesystem (btrfs, xfs, etx4, and so on) can be created on each partition.

There is a limit to the number of traditional PC-based partitions you can create. Originally only four partitions, referred to as *primary partitions*, were permitted. As more partitions were needed, a technique was created that makes it possible to convert one of the primary partitions

into an extended partition. Within an extended partition, you can create additional partitions called *logical partitions*.

In Figure 3.1, **/dev/sda1**, **/dev/sda2**, and **/dev/sda3** are primary partitions. The **/dev/sda4** partition is an extended partition that is used as a container for the **/dev/sda5**, **/dev/sda6**, and **/dev/sda7** logical partitions.

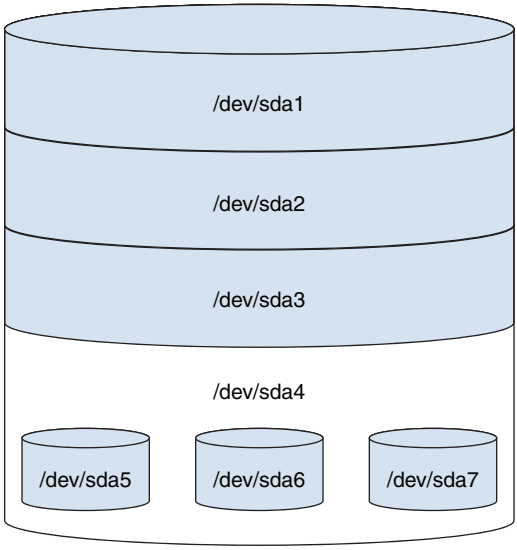

FIGURE 3.1 **Traditional Partition Structure**

On most Linux distributions that use traditional partitions, you are limited to a total of 15 partitions (though a kernel tweak can increase this number to 63).

Traditional partition tables are stored on the master boot record (MBR). A newer partition table, called the GUID partition table (GPT), does not face the same limitations or have the same layout as an MBR partition table.

Several different tools can be used to create or view partitions, including **fdisk**, **parted**, and the GUI-based tool provided by the installation program (which can vary based on the distribution).

Both **fdisk** and **parted** support command-line options, and both of them can be executed as interactive tools.

fdisk

The **fdisk** utility is an interactive tool that allows you to display and modify traditional (non-GUID) partition tables. To display a partition table, use the **-l** option (as the root user), like so:

```
# fdisk -l /dev/sda

Disk /dev/sda: 42.9 GB, 42949672960 bytes
4 heads, 32 sectors/track, 655360 cylinders, total 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sudo
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000c566f

      Device Boot Start End      Blocks    Id System
     /dev/sda1   * 2048 83886079 41942016  83 Linux
```

To modify the partition table of a drive, run the **fdisk** command without the **-l** option, as shown here:

```
# fdisk /dev/sda
Command (m for help):
```

There are several useful commands you can type at the **Command** prompt, including those listed in Table 3.1.

TABLE 3.1 **Partition-Related Commands**

Command	Description
d	Deletes a partition.
l	Lists partition types.
m	Prints a menu of possible commands.
n	Creates a new partition.
p	Prints the current partition table.
q	Quits without saving any changes.
t	Changes a partition table type.
w	Writes (saves) changes to the partition table on the hard drive.

parted

The **parted** utility is an interactive tool that allows you to display and modify both traditional and GUID partition tables. It can also create a filesystem on a partition.

To display a partition table, use the **-l** option and run the **parted** command as the root user, as in this example:

```
# parted -l /dev/sda
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 42.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number Start   End    Size    Type File system Flags
 1      1049kB 42.9GB primary ext4     boot


Model: Linux device-mapper (thin) (dm)
Disk /dev/mapper/docker-8:1-264916-
f9bd50927a44b83330c036684911b54e494e4e48efbc2329262b6f0e909e3d7d:
107GB
Sector size (logical/physical): 512B/512B
Partition Table: loop

Number Start   End   Size File system Flags
 1       0.00B 107GB ext4


Model: Linux device-mapper (thin) (dm)
Disk /dev/mapper/docker-8:1-264916-
77a4c5c2f607aa6b31a37280ac39a657bfd7ece1d940e50507fb0c128c220f7a:
107GB
Sector size (logical/physical): 512B/512B
Partition Table: loop


Number Start   End   Size File system Flags
 1      0.00B  107GB ext4
```

To modify the partition table of a drive, run the **parted** command without the **-l** option, as shown here:

```
# parted /dev/sda
GNU Parted 2.3
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

Table 3.2 shows several useful commands you can type at the **(parted)** prompt.

TABLE 3.2 **Partition-Related Commands Entered at the (parted) Prompt**

Command	Description
rm	Deletes a partition.
? or **help**	Prints a menu of possible commands.
mkpart	Creates a new partition.
mkpartfs	Creates a new partition and filesystem.
print	Prints the current partition table.
quit	Quits without saving any changes.
w	Writes (saves) changes to the partition table on the hard drive.

ExamAlert

Remember that **parted** can manage GUID, but **fdisk** cannot.

partprobe

The **partprobe** command is normally needed only in situations where the partition table has changed and the system needs to be informed of the changes. The most common example is when you use the **fdisk** command to change a partition on a device that currently has mounted filesystems. The **fdisk** command attempts to inform the system of changes to the partition table by using a kernel call, which fails because of the "live" filesystem. To overcome this, just execute the **partprobe** command after exiting the **fdisk** utility.

Mounting Local and Remote Devices

Mounting is the process of making storage devices available to the Linux filesystem. This section focuses on the files and tools that are used to manage the mounting process.

systemd.mount

The **systemd.mount** configuration is used by Systemd to mount and unmount resources during the boot process. It utilizes the **/etc/fstab** file (which is described next). To learn more about Systemd, see Chapter 4, "Configure and Use the Appropriate Processes and Services."

/etc/fstab

The **/etc/fstab** file is used to specify which filesystems to mount, where to mount the filesystems, and what options to use during the mount process. This file is used during the boot process to configure filesystems to mount on bootup.

Each line of the **/etc/fstab** file describes one mount process. The following is an example of one of these lines:

```
/dev/sda1 / ext4 defaults 1 1
```

Each line is broken into six fields of data, separated by whitespace:

- The device to mount (in the preceding example, **/dev/sda1**).
- The mount point (**/**).
- The filesystem type (**ext4**).
- The mount options (**defaults**).
- Dump level (**1**). This field is related to the **dump** command and is rarely used.
- The **fsck** pass field (**1**). The value **0** means "do not run **fsck** on this filesystem during system boot," whereas a value of **1** or higher means "run **fsck** on this filesystem during system boot."

Note that a complete **/etc/fstab** file contains multiple entries as well as some comments that provide basic documentation. Here is an example:

```
#
# /etc/fstab
# Created by anaconda on Tue Jun 22 03:22:27 2021
#
# Accessible filesystems, by reference, are maintained under '/dev/
disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more
info.
#
# After editing this file, run 'systemctl daemon-reload' to update
systemd
# units generated from this file.
#
```

```
/dev/mapper/cl-root     /               xfs     defaults        0 0
UUID=abc23ff6-7841-463b-bdac-45b3ae868cee /boot      xfs
defaults        0 0
/dev/mapper/cl-home     /home           xfs     defaults        0 0
/dev/mapper/cl-swap     none            swap    defaults        0 0
```

ExamAlert

It is a good idea to memorize the different fields of the **/etc/fstab** file because the Linux+ XK0-005 exam typically includes questions on the contents of this file.

mount

The **mount** command can display the currently mounted filesystems, as shown in this example:

```
# mount
/dev/sda1 on / type ext4 (rw)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs
 (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs
 (rw,noexec,nosuid,nodev,size=104857600, mode=0755)
none on /sys/fs/pstore type pstore (rw)
rpc_pipefs on /run/rpc_pipefs type rpc_pipefs (rw)
systemd on /sys/fs/cgroup/systemd type cgroup
 (rw,noexec,nosuid,nodev, none,name=systemd)
```

The **mount** command can also be used to manually mount a filesystem. Provide the device to mount as the first argument and the mount point (that is, mount directory) as the second argument and execute the following commands as the root user:

```
# mkdir /data
# mount /dev/sdb1 /data
```

Table 3.3 details important options for the **mount** command.

TABLE 3.3 **mount Command Options**

Option	Description
-a	Mounts all filesystems listed in the **/etc/fstab** file that have the mount option **auto**.
-o	Specifies a mount option (for example, **mount -o acl /dev/sdb1 /data**).
-t	Specifies the filesystem type to mount. This is typically not necessary because the **mount** command can determine the filesystem type by probing the partition.

Use the **umount** command to manually unmount a filesystem:

```
# mount | grep /data
/dev/sdb1 on /data type ext3 (rw)
# umount /data
```

Table 3.4 details important options for the **umount** command.

TABLE 3.4 **umount Command Options**

Option	Description
-r	Attempts to mount the filesystem as read-only if the unmount operation fails.
-f	Forces the unmount. This is typically used on NFS mounts when the NFS server is nonresponsive.

If you have just created the filesystem, it will likely be easy to remember which device file was used to access the filesystem. However, if you forget which device files are available, you can execute the **lsblk** command, as shown here:

```
# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    252:0    0 254G 0  disk
| vda1  252:1  0 250G 0 part /
vda2    252:2  0   4G 0 part [SWAP]
```

This command was performed on a native virtual machine—hence the device names **vda**, **vda1**, and **vda2**.

You can see your label and UUIDs by using the **blkid** command:

```
# blkid
/dev/sda1: UUID="4d2b8b91-9666-49cc-a23a-1a183ccd2150" TYPE="ext4"
/dev/sda3: LABEL="mars" UUID="bab04315-389d-42bf-
  9efa-b25c2f39b7a0" TYPE="ext4"
/dev/sda4: UUID="18d6e8bc-14a0-44a0-b82b-e69b4469b0ad" TYPE="ext4"
```

Linux Unified Key Setup (LUKS)

An important technology related to Linux filesystem encryption is a specification called LUKS (Linux Unified Key Setup). As a specification, LUKS describes how filesystems are to be encrypted on Linux. It does not provide any software, and it is not an official standard (although specifications are commonly referred to as "unofficial standards").

Because it is a specification, LUKS does not force you to use any one specific software tool to encrypt a filesystem. Different tools are available, but for purposes of demonstration, this chapter shows a kernel-based implementation called DMCrypt.

DMCrypt is a kernel module that allows the kernel to understand encrypted filesystems. In addition to the DMCrypt module, you should be aware of two commands that you can use to create and mount an encrypted filesystem: **cryptsetup** and **cryptmount**. Note that you would use only one of these two commands (most likely **cryptsetup**) to configure an encrypted filesystem.

This section demonstrates how to create a new encrypted filesystem by using the **cryptsetup** command. To begin, you may need to load some kernel modules:

```
[root@onecoursesource ~]# modprobe dm-crypt
[root@onecoursesource ~]# modprobe aes
[root@onecoursesource ~]# modprobe sha256
```

Next, create a LUKS-formatted password on a new partition. Note that if you are using an existing partition, you first need to back up all data and unmount the partition. The following command overrides data on the **/dev/sda3** partition:

```
[root@onecoursesource ~]# cryptsetup --verbose --verify-passphrase
luksFormat /dev/sda3
WARNING!
========
This will overwrite data on /dev/sda3 irrevocably.
Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:
Command successful.
```

Notice from this output of the **cryptsetup** command that you are prompted to provide a passphrase (a string of characters, such as a sentence or simple phrase). You will need to use this passphrase to decrypt the filesystem whenever you need to mount the filesystem.

Filesystem Management

Several sets of tools are used to manage filesystems and storage devices. This section covers the XFS, EXT, and Btrfs tools.

XFS Tools

XFS is a filesystem designed for high performance and for handling larger file sizes.

The **xfs_metadump** command dumps (copies) metadata from an unmounted XFS filesystem into a file to be used for debugging purposes. Table 3.5 details some important options for XFS tools.

TABLE 3.5 **XFS Tool Options**

Option	Description
-e	Stops the dump if a filesystem error is found.
-g	Shows the progress of the dump.
-w	Displays error messages if filesystem errors occur.

The **xfs_info** command is used to display the geometry of an XFS filesystem, similarly to the **dumpe2fs** command for ext2/ext3/ext4 filesystems. There are no special options for the **xfs_info** command.

ext4 Tools

The ext4 filesystem is a replacement for the ext3 filesystem. It supports larger filesystems and individual file sizes. ext4 provides better performance than ext3.

The **mkfs** command creates a filesystem on a partition. The basic syntax of the command is **mkfs -t** *fstype partition*, where *fstype* can be one of the types described in Table 3.6.

TABLE 3.6 **fstype Options**

Type	Description
ext2	Creates an ext2 filesystem (the default on most distributions).
ext3	Creates an ext3 filesystem.
ext4	Creates an ext4 filesystem.
bfs	Creates a btrfs filesystem.
vfat	Creates a VFAT (DOS) filesystem.
ntfs	Creates an NTFS (Windows) filesystem.
xfs	Creates an XFS filesystem.

Note that the **mkfs** command is a front-end utility to other commands. For example, if you run the command **mkfs -t ext4 /dev/sdb7**, the **mkfs.ext4 /dev/sdb7** command will actually be executed.

Each specific filesystem-creation utility has dozens of possible options that affect how the filesystem is created. These options are passed from the **mkfs** command to the specific filesystem-creation command that **mkfs** launches.

The **resize2fs** command is commonly used in conjunction with resizing a logical volume. Once the LV has been resized, the underlying ext3 or ext4 filesystem also must be resized.

If the plan is to make the LV larger, the **lvextend** command should be executed first, followed by the **resize2fs** command. No size value is needed for the **resize2fs** command, as it increases to the size of the LV. Here is an example:

```
lvextend -L+1G /dev/vol0/lv0
resize2fs /dev/vol0/lv0
```

If the plan is to make the LV smaller, you first have to resize the filesystem and then use the **lvreduce** command to decrease the size of the LV. If you reduced the LV first, the system would not be able to access the filesystem beyond the new LV size:

```
resize2fs /dev/vol0/lv0 2G
lvreduce -L2G /dev/vol0/lv0
```

The **fsck** utility is designed to find filesystem problems on unmounted filesystems. For example, you could run this command as the root user:

```
# fsck /dev/sdb1
fsck from util-linux 2.20.1
e2fsck 1.42.9 (4-Feb-2014)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sdb1: 11/12544 files (0.0% non-contiguous), 6498/50176 blocks
```

This utility is fairly straightforward. It calls the correct filesystem check utility based on a probe of the filesystem and then prompts the user when errors are found. To fix an error, answer **y** or **yes** to each of the prompts. (Because **yes** is almost always the appropriate answer, the **fsck** utility supports a **-y** option, which automatically answers **yes** to each prompt.)

The **fsck** command executes filesystem-specific utilities. In the case of ext2/ext3/ext4 filesystems, the **fsck** command executes the **e2fsck** utility. See the "**fsck**" section, earlier in this chapter, for details regarding the **fsck** command.

The **tune2fs** command is used to display or modify specific metadata for an ext2/ext3/ext4 filesystem. For example, by default, 5% of an ext2/ext3/ext4 filesystem is reserved for the system administrator. You can run the following command as the root user:

```
# tune2fs -l /dev/sdb1 | grep block
Inode count:           12544
Block count:           50176
Reserved block count: 2508
Mount count:             0
Maximum mount count:  -1
```

Note that the reserved block count (2508) is 5% of the block count (50176). Use the following command to change this to a different percentage:

```
# tune2fs -m 20 /dev/sdb1
tune2fs 1.42.9 (4-Feb-2014)
Setting reserved blocks percentage to 20% (10035 blocks)
```

Table 3.7 details important options for the **tune2fs** command.

TABLE 3.7 **tune2fs Options**

Option	Description
-J	Modifies journal options.
-o	Modifies mount options.
-L	Modifies the filesystem label.
-l	Lists filesystem metadata.
-m	Modifies the percentage of the filesystem reserved for the root user.

To change the label of a filesystem, use the **e2label** command:

```
# e2label /dev/sda3 pluto
# blkid
/dev/sda1: UUID="4d2b8b91-9666-49cc-a23a-1a183ccd2150" TYPE="ext4"
/dev/sda3: LABEL="pluto" UUID="bab04315-389d-42bf-
 9efa-b25c2f39b7a0" TYPE="ext4"
/dev/sda4: UUID="18d6e8bc-14a0-44a0-b82b-e69b4469b0ad" TYPE="ext4"
```

Btrfs Tools

Btrfs, known as "butter FS," is a general-purpose Linux filesystem that uses a method called "B-trees" to manage the filesystem structure. To manage a Btrfs filesystem, use the **btrfs** utility.

A large number of subcommands are available with the **btrfs** utility. Table 3.8 details some of the important subcommands for the **btrfs** utility.

TABLE 3.8 **btrfs Utility Subcommands**

Subcommand	Description
filesystem	Manages filesystem tasks and displays filesystem information.
subvolume	Manages subvolumes and displays subvolume information.
rescue	Performs filesystem rescue (fix) operations.

You can use a subcommand as an argument to the **btrfs** utility. For example, the following would display filesystem information on the Btrfs filesystem on all devices:

```
btrfs filesystem show
```

Monitoring Storage Space and Disk Usage

Gathering storage space information can be useful when determining if there is enough space available to install a software package or database. It can also be helpful in determining which directories contain files that are using a large amount of space. The commands described in this section provide you with information needed in monitoring storage space and disk usage.

df

The **df** command displays usage of partitions and logical devices:

```
# df
Filesystem   1K-blocks   Used     Available Use% Mounted on
udev          2019204    12       2019192     1% /dev
tmpfs          404832    412       404420      1% /run
/dev/sda1    41251136    6992272  32522952   18% /
none         4           0         4           0% /sys/fs/cgroup
none         5120        0        5120        0% /run/lock
none         2024144     0        2024144     0% /run/shm
none         102400      0         102400     0% /run/user
```

Table 3.9 details important options for the **df** command.

TABLE 3.9 **df Command Options**

Option	Description
-h	Displays values in human-readable size.
-i	Displays inode information.

du

The **du** command provides an estimated amount of disk space usage in a directory structure. For example, the following command displays the amount of space used in the **/usr/lib** directory:

```
# du -sh /usr/lib
791M /usr/lib
```

Table 3.10 details important options for the **du** command.

TABLE 3.10 **du Command Options**

Option	Description
-h	Displays values in a human-readable size. (Instead of always displaying in bytes, it displays in more understandable values, such as megabytes or kilobytes, depending on the overall size of the file.)
-s	Displays a summary rather than the size of each subdirectory.

Creating and Modifying Volumes Using Logical Volume Manager (LVM)

LVM is designed to address a few issues with regular partitions, including the following:

- Regular partitions are not resizable. LVM provides the means to change the size of partition-like structures called *logical volumes*.
- The size of a regular partition cannot exceed the overall size of the hard disk on which the partition is placed. With LVM, several physical devices can be merged together to create a much larger logical volume.
- Active filesystems pose a challenge when you're backing up data because changes to the filesystem during the backup process could result in corrupt backups. LVM provides a feature called a "snapshot" that makes it easy to correctly back up a live filesystem.

LVM consists of one or more physical devices merged into a single container of space that can be used to create partition-like devices. The physical devices can be entire hard disks, partitions on a hard disk, removable media devices (USB drives), software RAID devices, or any other storage devices.

ExamAlert

LVM can be a challenging topic. For the Linux+ XK0-005 exam, make sure you understand the differences between physical volumes (PVs), volume groups (VGs), and logical volumes (LVs).

pvs

You can display PVs by using the **pvs** command, as demonstrated in the following example:

```
# pvs
  PV         VG    Fmt  Attr PSize  PFree
  /dev/sdb1  my_vg lvm2 a-   19.24G 18.14G
  /dev/sdc1  my_vg lvm2 a-   19.24G 18.06G
  /dev/sdd1  my_vg lvm2 a-   19.24G 18.14G
```

See the "**pvcreate**" section, later in this chapter, for more information about PVs.

vgs

You can display VGs by using the **vgs** command, as demonstrated in the following example:

```
# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  Vol00         3   2   0   wz--n- 33.22G 0
  Vol01         2   1   0   wz--nc 47.00G 8.00M
```

See the "**vgcreate**" section, later in this chapter, to learn more about VGs.

lvs

You can display LVs by using the **lvs** command, as demonstrated in the following example:

```
# lvs
  LV      VG        Attr     LSize  Pool Origin Data%  Move Log Copy%
Convert
  lv_root Vol00    -wi-ao--  12.22g
  lv_swap Vol00    -wi-ao--  3.12g
```

See the "**lvcreate**" section, later in this chapter, to learn more about LVs.

lvchange

The **lvchange** command allows you to change the attributes of an LV. One common use of this command is to change the state of an LV to active:

```
lvchange -ay /dev/vol0/lv0
```

The same command can be used to deactivate an LV:

```
lvchange -an /dev/vol0/lv0
```

lvcreate

The space within PVs is broken into small chunks called *extents*. Each extent is 4MB by default (but can be modified when creating the VG by using the **-s** option to the **vgcreate** command). To create an LV, execute the **lvcreate** command and either specify how many extents to assign to the LV or provide a size (which will be rounded up to an extent size), as shown here:

```
lvcreate -n lv0 -L 400MB vol0
```

The result of this command will be a device file named **/dev/vol0/lv0** that will have 400MB of raw space available. See Figure 3.2 for a visual example.

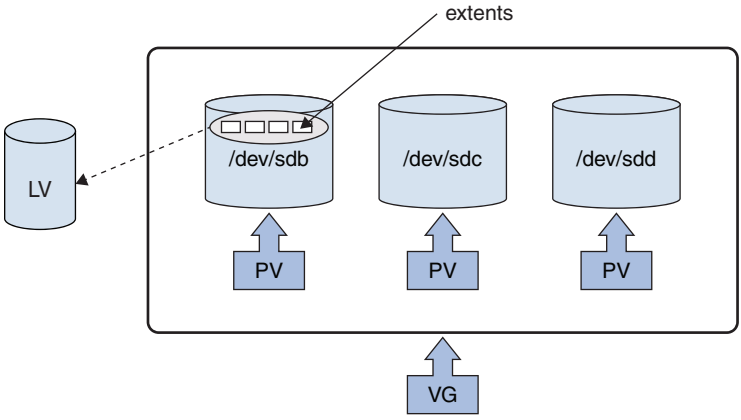

FIGURE 3.2 **Logical Volumes**

vgcreate

After creating PVs with the **pvcreate** command (see the "**pvcreate**" section, later in this chapter), place these PVs into a VG by executing the following command:

```
# vgcreate vol0 /dev/sdb /dev/sdc /dev/sdd
```

Consider a VG to be a collection of storage devices that you want to use to create partition-like structures called logical volumes (LVs). So, if **/dev/sdb** is a 60GB hard drive, **/dev/sdc** is a 30GB hard drive, and **/dev/sdd** is a 20GB hard drive, then the VG created by the previous command has 110GB of space available to create the LVs. You could create a single LV using all 110GB or many smaller LVs. See Figure 3.3 for a visual demonstration of volume groups.

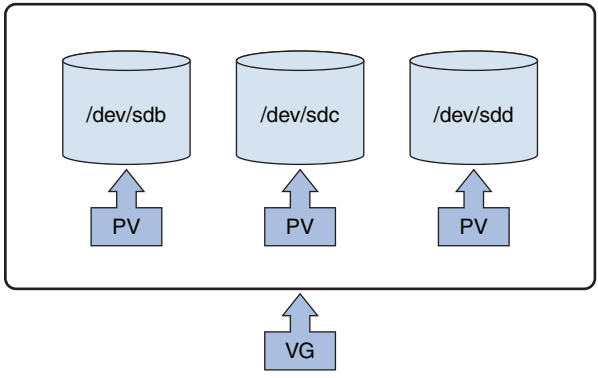

FIGURE 3.3 **Volume Groups**

lvresize

The **lvresize** command allows you to change the size of an LV. For example, the following command adds 10GB to the **/dev/vol0/lv0** LV, assuming that there is enough free space in the VG:

```
lvresize -L +10G dev/vol0/lv0
```

pvcreate

The first step in creating an LVM is to convert existing physical devices into PVs. This is accomplished by executing the **pvcreate** command. For example, if you have three hard drives, as shown in Figure 3.4, and you want to make them all PVs, you can execute the following command:

```
pvcreate /dev/sdb /dev/sdc /dev/sdd
```

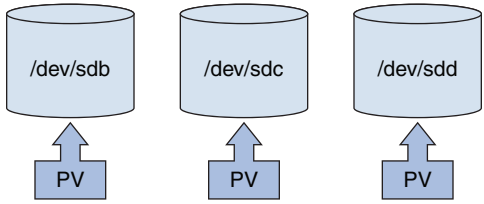

FIGURE 3.4 **Physical Volumes**

vgextend

When you want to add a new PV to a VG, you can use the **vgextend** command. For example, the following command adds the **/dev/sde** PV to the **vol0** VG:

```
vgextend vol0 /dev/sde
```

> **Note**
>
> The **/dev/sde** device must first be configured as a PV using the **pvcreate** command. See the **"pvcreate"** section, later in this chapter, for more information.

Inspecting RAID Implementations

A RAID device is used to provide redundancy. Two or more physical devices can be combined to create a single device that stores data in a way that mitigates data loss in the event that one of the physical storage devices fails.

There are several different types of RAID devices, called "RAID levels." While you can create many different types of software RAID levels, only a few of the levels really make sense for software RAID devices. When you configure a software RAID device, you should consider the RAID levels described in Table 3.11.

TABLE 3.11 **RAID Levels**

Level	Description
RAID 0	RAID 0 involves writing to multiple drives as if they were a single device. The writes are performed using "striping," in which some data is written to the first drive, then some data is written to the second drive, and so on. RAID 0 combines multiple smaller hard disk drives into a single storage space. So, if you have three 20GB hard disks, you could merge them together into a single 60GB storage device. Software RAID 0 is extremely rare for a couple of reasons. One reason is that this RAID level provides no redundancy (which makes it strange that it is considered RAID). Another reason is that LVM (Logical Volume Manager) also makes it possible to merge multiple devices together, and LVM has several advantages over software RAID 0 that make it the better choice. (LVM is discussed earlier in this chapter.)
RAID 1	With RAID 1, also called "mirroring," two or more disk drives appear to be a single storage device. Data that is written to one disk drive is also written to all of the others. If one drive fails, the data is still available on the other drives. Software RAID 1 is very popular because hard drives are fairly cheap, and the redundancy provided by RAID 1 limits data loss.
RAID 4	Implementing RAID 4 requires at least three drives. All but one drive is used to store filesystem data, and the last drive is used to store parity data—that is, a value derived from the data stored on the other devices in the RAID. Suppose there are three devices in the RAID; two of them (called devices A and B in this example) store regular filesystem data, and one (called device C) stores the parity data. If device A fails, then the data that it stored could be rebuilt using a comparison of the data in device B and the parity data in device C. How does this work? Consider the following formula: 1 + 1 = 2 (or, perhaps, A + B = C). If someone removed one of the values in that formula (for example, ___ + 1 = 2), you could recover that information by comparing the other two values. This is the basic idea of parity data.
RAID 5	RAID 5 is very similar to RAID 4. The difference is that RAID 5 doesn't use a single parity disk but rather spreads the parity data over multiple disks in a "round robin" approach.

Level	Description
RAID 10	Also called RAID 1+0, this RAID level combines the advantages of both RAID 1 and RAID 0. First, two or more sets of two devices are placed into multiple RAID 1 devices. This provides redundancy. Then they are merged together into a RAID 0 device to create a much larger storage container. In terms of software RAID, this RAID level is fairly rare, mostly because of the advantages of LVM over RAID 0. A much more common scenario would be a RAID 0 plus LVM combination. For hardware RAID, this level is not uncommon.

ExamAlert

Know these RAID levels for the Linux+ XK0-005 exam! The exam is likely to ask you to specify which RAID level you should use in a given scenario.

mdadm

To create a software RAID device, execute a command like the following:

```
# mdadm -C /dev/md0 -l 1 -n 2 /dev/sdb /dev/sdc
mdadm: array /dev/md0 started.
```

This command uses the following options:

- **-C:** Specifies the device name for the RAID device.
- **-l:** Specifies the RAID level.
- **-n:** Specifies the number of physical storage devices in the RAID array.

/proc/mdstat

After you create a RAID device, you can see information about the device by viewing the contents of the **/proc/mdstat** file, as in this example:

```
# more /proc/mdstat
Personalities : [raid1]
md0 : active raid1 sdc[1] sdb[0]
      987840 blocks [2/2] [UU]
unused devices: <none>
```

The **mdadm --detail** command can also be useful for displaying information about a software RAID device.

Storage Area Network (SAN)/ Network-Attached Storage (NAS)

This section provides information about SAN and NAS devices.

multipathd

Some storage devices are available only through the network. This creates a point of failure: the network itself. If you lose network access to a remote storage device, perhaps because a router went down or a new firewall rule was implemented, applications on your system might fail to function properly.

Multipathing involves creating different network paths to a remote storage device. This requires additional network setup, including configuring different routes to the network storage device. The multipath daemon (**multipathd**) is the process that manages these network paths.

> **ExamAlert**
>
> Details regarding configuring multipathing are beyond the scope of the Linux+ XK0-005 exam and are not covered in this book. However, multipathing is included as an exam objective, so be prepared to answer questions regarding the purpose of multipathing.

Network Filesystems

As with most other operating systems, Linux allows you to access filesystems that are shared across the network. This section describes two of the most common network filesystems for Linux: NFS and SMB.

Network File System (NFS)

Network File System (NFS) is a Distributed File System (DFS) protocol that has been in use for more than 40 years. NFS was originally created by Sun Microsystems in 1984 to provide an easy way for administrators to share files and directories from one Unix system to another.

Since its inception, NFS has been ported to several different operating systems, including Linux and Microsoft Windows. While it might not be as popular as SAMBA, there are many organizations that still use NFS to share files.

The primary configuration for the NFS server is the **/etc/exports** file. This is the file you use to specify what directories you want to share to the NFS clients. The syntax of this file is as follows:

```
Directory    hostname(options)
```

Directory should be replaced with the name of the directory that you want to share (for example, **/usr/share/doc**), and *hostname* should be a client hostname that can be resolved into an IP address. The *options* value is used to specify how the resource should be shared.

For example, the following entry in the **/etc/exports** file would share the **/usr/share/doc** directory to the NFS client **jupiter** as read/write and to the NFS client mars as read-only:

```
/usr/share/doc jupiter(rw) mars(ro)
```

Note that there is a space between **jupiter** and **mars**, but there is no space between each hostname and its corresponding option. A common mistake of novice administrators is to provide an entry like the following:

```
/usr/share/doc jupiter (rw)
```

This line would share the **/usr/share/doc** directory to the **jupiter** host with the default options, and all other hosts would have read/write access to this share.

When specifying a hostname in the **/etc/exports** file, the following methods are permitted:

- **hostname:** A hostname that can be resolved to an IP address.
- **netgroup:** An NIS netgroup using the designation @*groupname*.
- **domain:** A domain name using wildcards. For example, ***.onecoursesource.com** would include any machine in the onecoursesource.com domain.
- **Network:** A network defined by IP addresses using either VLSM (variable-length subnet masking) or CIDR (classless interdomain routing). Examples include 192.168.1.0/255.255.255.0 and 192.168.1.0/24.

There are many different NFS sharing options, including

- **rw:** Shares as read/write. Keep in mind that normal Linux permissions still apply. Important: This is a default option.
- **ro:** Shares as read-only.

- **sync:** Makes file data changes to disk immediately. This has an impact on performance but is less likely to result in data loss. On some distributions, this is the default.
- **async:** The opposite of sync. Initially makes file data changes to memory. This speeds up performance but is more likely to result in data loss. On some distributions, this is the default.
- **root_squash:** Maps the root user and group account from the NFS client to the anonymous accounts, typically either the nobody account or the nfsnobody account. Important: This is a default option.
- **no_root_squash:** Maps the root user and group account from the NFS client to the local root and group accounts.

Server Message Block(SMB)/Common Internet File System (CIFS)

One of the ways to share files between different systems is by using a protocol called SMB (Server Message Block). This protocol was invented in the mid-1980s by IBM to make it possible to share directories between hosts on a local area network (LAN). Distributed File System (DFS) is used to share files and directories across a network. In addition to SMB, Network File System (NFS) is a popular DFS for Linux. (NFS is covered earlier in this chapter.)

You may often hear the acronym CIFS (Common Internet File System) used in conjunction with SMB. CIFS is an SMB-based protocol that is popular on Microsoft Windows systems. Typically, the two abbreviations are used interchangeably (or together, as SMB/CIFS), but there are subtle differences between these protocols. This book uses the term SMB from this point on.

You can also share printers using SMB as well as share files between different operating system types. In fact, one common SMB task is to share printers between Linux and Microsoft Windows systems.

The Linux-based software that allows SMB sharing is called SAMBA. The configuration file for SAMBA is the **/etc/SAMBA/smb.conf** file.

To give you an idea of what a typical **smb.conf** file looks like, examine the following output, which demonstrates a typical default **smb.conf** file with all comment and blank lines removed:

```
[root@onecoursesource ~]# grep -v "#" /etc/SAMBA/smb.conf | grep -v
";" | grep -v "^$"
[global]
```

```
        workgroup = MYGROUP
        server string = SAMBA Server Version %v
        security = user
        passdb backend = tdbsam
        load printers = yes
        cups options = raw
[homes]
        comment = Home Directories
        browseable = no
        writable = yes
[printers]
        comment = All Printers
        path = /var/spool/SAMBA
        browseable = no
        guest ok = no
        writable = no
        printable = yes
```

Table 3.12 describes the common options for the SAMBA configuration file.

TABLE 3.12 **SAMBA Configuration File Options**

Option	Description
workgroup	This is the NetBIOS (Network Basic Input/Output System) workgroup or NetBIOS domain name. It enables you to group together a set of machines, which may be important if you want to communicate with Microsoft Windows systems via SMB.
server string	This is a description of the server and is useful when a remote system is attempting to connect to the server to determine what services the server provides. The value **%v** is replaced with SAMBA's version number. The value **%h** can be used to symbolize the server's hostname.
security	This determines what type of user authentication method is used. The value **user** means SAMBA user accounts will be used. If you have a DC (domain controller) on a Microsoft Windows system, you specify the domain. To authenticate via Active Directory, you specify **ads**.
passdb backend	This specifies how the SAMBA account data is stored. It is not typically something you change unless you are an expert.

Option	Description
load printers	This option, if set to **yes**, tells SAMBA to share all CUPS (Common UNIX Printing System, a common printing protocol in Linux and Unix) printers by default. While this can be handy, you don't always want to share all CUPS printers on the SAMBA server. Individual printer shares can be handled in separate sections. Note that for SAMBA to be able to load all CUPS printers automatically, the **[printers]** section needs to be properly configured.
cups options	These options regarding CUPS are modified only by experts.

Storage Hardware

Linux provides several commands that you can use to display information about storage hardware. This section covers the most commonly used of these commands.

lsscsi

A Small Computer System Interface (SCSI) device is a storage device (typically a hard drive, though it can also be a tape drive). These devices are fairly rare on Intel-based PCs, which is what most Linux systems are installed on. If your system is a SCSI device, you can list information about it by using the **lsscsi** command.

lsblk

If you have just created the filesystem, it will likely be easy to remember which device file was used to access the filesystem. However, if you forget which device files are available, you can execute the **lsblk** command. The following is an example of this command performed on a native virtual machine—hence the device names **vda**, **vda1**, and **vda2**:

```
# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    252:0    0 254G 0  disk
| vda1  252:1  0 250G 0 part /
vda2    252:2  0   4G 0 part [SWAP]
```

blkid

You can see your label and UUIDs by using the **blkid** command:

```
# blkid
/dev/sda1: UUID="4d2b8b91-9666-49cc-a23a-1a183ccd2150" TYPE="ext4"
/dev/sda3: LABEL="mars" UUID="bab04315-389d-42bf-
  9efa-b25c2f39b7a0" TYPE="ext4"
/dev/sda4: UUID="18d6e8bc-14a0-44a0-b82b-e69b4469b0ad" TYPE="ext4"
```

fcstat

You can display information about storage devices that are attached to Fibre Channel by using the **fcstat** command. Fibre Channel storage devices are fairly rare for Linux devices, but you should know the common options for this command, as listed in Table 3.13.

TABLE 3.13 **fcstat Command Options**

Option	Description
link_stats	Displays link information in the event of connection errors.
device_map	Displays information about attached devices.

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. Which utility can you use to create partitions on a GUID partition table?
   - ❍ **A. partprobe**
   - ❍ **B. parted**
   - ❍ **C. fdisk**
   - ❍ **D. mkfs**

2. Which field in the following example denotes the mount point?

   ```
   /dev/sda1 / ext4 defaults 1 1
   ```

   - ❍ **A. /dev/sda1**
   - ❍ **B. /**
   - ❍ **C. ext4**
   - ❍ **D. defaults**

3. Which command allows you to add a new PV to a VG?
   - ❍ **A. pvadd**
   - ❍ **B. pvextend**
   - ❍ **C. vgadd**
   - ❍ **D. vgextend**

4. Which command can be used to view the labels and UUIDs of devices?
   - ❍ **A. blkid**
   - ❍ **B. lsblk**
   - ❍ **C. lsscsi**
   - ❍ **D. fcstat**

Cram Quiz Answers

1. **B. parted** allows you to modify both MBR and GUID partition tables. The **fdisk** utility only allows you to modify MBR partition tables. The other answers are not partition tools.
2. **B.** The second field of the **/etc/fstab** file contains the mount point.
3. **D.** The **vgextend** command allows you to add a new PV to an existing VG, as in this example: **vgextend vol0 /dev/sde**. The rest of the answers are not valid commands.
4. **A.** The **blkid** command is used to display UUIDs and labels on storage devices.

CHAPTER 4

Configure and Use the Appropriate Processes and Services

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.4:** Given a scenario, configure and use the appropriate processes and services.

On Linux systems, a service is a feature of the operating system or software that acts like a server. Administrators need to know how to manage these services by using the **systemctl** command, which is covered in this chapter.

Administrators often need to use the **crontab** and **at** utilities, which make it possible to schedule processes (or programs) in the future. Speaking of processes, it is also important to know how to list and control processes. These topics are also covered in this chapter.

This chapter provides information on the following topics: system services, scheduling of services, and process management.

System Services

Systemd is a feature of Linux that is used to control which services are started during the boot process. Systemd uses "targets," and each target has specific services that start. Figure 4.1 shows an example of a typical Systemd boot sequence.

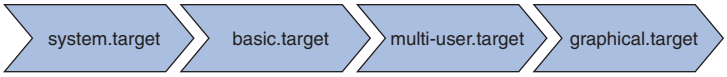

FIGURE 4.1 **Overview of the Systemd Boot Process**

Targets are defined in the **/usr/lib/systemd/system** directory. Consider the following example of a target file:

```
# cat /usr/lib/systemd/system/graphical.target
# This file is part of systemd.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License
 as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target
 display-manager.service
AllowIsolate=yes
```

The default target is defined by a symbolic link from **/etc/systemd/system/default.target** to the target in the **/usr/lib/systemd/system** directory, as in this example:

```
# ls -l /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 36 Jun 11 20:47
 /etc/systemd/system/default.target ->
 /lib/systemd/system/graphical.target
```

Use the following command to display the default target:

```
# systemctl get-default
multi-user.target
```

Use the **systemctl list-unit-files --type=target** command to list the available targets. This command provides a large amount of output, and the following example uses the **head** command to limit the output:

```
# systemctl list-unit-files --type=target | head
UNIT FILE STATE
basic.target              static
bluetooth.target          static
cryptsetup-pre.target     static
cryptsetup.target         static
ctrl-alt-del.target       disabled
cvs.target                static
default.target            enabled
emergency.target          static
final.target              static
```

Use the following command to set the default target:

```
# systemctl set-default graphical-user.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/graphical-
 user.target' '/etc/systemd/system/default.target'
```

systemctl

The **systemctl** command is used to administer a Systemd-based distribution. For example, to change to another target, execute the following command:

```
systemctl isolate multi-user.target
```

stop

The **stop** option is used with the **systemctl** command to stop a service that is currently running.

Syntax:

```
systemctl stop service
```

You can determine whether a service is currently running by using the command as follows:

```
# systemctl active cups
enabled
```

start

The **start** option is used with the **systemctl** command to start a service that is not currently running.

Syntax:

```
systemctl start service
```

You can determine whether a service is currently running by using the command as follows:

```
# systemctl active cups
enabled
```

restart

The **restart** option is used with the **systemctl** command to restart a service that is currently running.

Syntax:

```
systemctl restart process_name
```

You can determine whether a service is currently running by using the command as follows:

```
# systemctl active cups
enabled
```

status

The **status** option is used with the **systemctl** command to display the current status of a service.

Syntax:

```
systemctl status process_name
```

Here is an example of displaying the status of the CUPS service:

```
# sysctl status cups
 cups.service - CUPS Scheduler
 Loaded: loaded (/lib/systemd/system/cups.service; enabled;
 vendor preset: enabled)
 Active: active (running) since Mon 2019-01-07 00:10:18 PST;20h ago
 Docs: man:cupsd(8)
 Main PID: 4195 (cupsd)
 Tasks: 1 (limit: 4780)
 CGroup: /system.slice/cups.service
          4195 /usr/sbin/cupsd -l

Jan 07 00:10:18 student-VirtualBox systemd[1]: Started CUPS Scheduler.
```

enable

The **enable** option is used with the **systemctl** command to start a service at boot time.

Syntax:

```
systemctl enable service
```

You can determine if a service is currently enabled or disabled by using the command as follows:

```
# systemctl is-enabled cups
enabled
```

disable

The **disable** option is used with the **systemctl** command to change a service this is currently started at boot time so that it won't start automatically then.

Syntax:

```
systemctl disable service
```

You can determine if a service is currently enabled or disabled by using the command as follows:

```
# systemctl is-enabled cups
enabled
```

mask

To mask a service is to make it completely impossible to start or enable. This is commonly done when there is a conflicting service on a system that, for some reason, can't or shouldn't be removed from the system.

Syntax to mask a service:

```
systemctl mask service
```

> **ExamAlert**
>
> Understand the difference between masking a service and disabling a service before you take the Linux+ XK0-005 exam.

Scheduling Services

Sometimes you need to be able to execute commands in the future. The **cron** and **at** utilities described in this section allow you to schedule jobs (that is, commands) to be executed in the future.

cron

The **cron** service allows you to schedule processes to run at specific times. This service makes use of the **crond** daemon, which checks every minute to see what processes should be executed. This daemon checks both **crontab** and **at** jobs to determine what commands to execute and when to execute them. See the "**crontab**" and "**at**" sections, later in this chapter, for further details.

crontab

The **crontab** command allows you to view or modify your **crontab** file. This file allows you to schedule a command to be executed regularly, such as once an hour or twice a month.

Table 4.1 lists some important options for the **crontab** command.

TABLE 4.1 **crontab Command Options**

Option	Description
-e	Edits the **crontab** file.
-l	Lists the **crontab** file.
-r	Removes all entries from the **crontab** file.

Each line of the **crontab** file is broken into fields, separated by one or more space characters. Table 4.2 describes these fields.

TABLE 4.2 **crontab File Fields**

Field	Description
First field: **Minute**	The minute when the command should execute. Values can be 0–59. You can use a single value, a list of values (such as 0,15,30,45), or a range of values (such as 1–15). You can use the * character to indicate "all possible values."
Second field: **Hour**	The hour when the command should execute. Values can be 0–23. You can use a single value, a list of values (such as 0,6,12,18), or a range of values (such as 8–16). You can use the * character to indicate "all possible values."
Third field: **Day of the Month**	The day of the month the command should execute. Values can be 1–31. You can use a single value, a list of values (such as 1,15), or a range of values (such as 1–10). You can use the * character to indicate "not specified" unless the fifth field is also an * character, in which case the * character means "all possible values."
Fourth field: **Month**	The month that the command should execute. Values can be 1–12. You can use a single value, a list of values (such as 6,12), or a range of values (such as 1–3). You can use the * character to indicate "all possible values."
Fifth field: **Day of the Week**	The day of the week the command should execute. Values can be 0–7 (where 0=Sunday, 1=Monday,…6=Saturday, 7=Sunday). You can use a single value, a list of values (such as 1,3,5), or a range of values (such as 1–5). You can use the * character to indicate "not specified" unless the fifth field is also an * character, in which case the * character means "all possible values."
Sixth field: **Command Name**	The name of the command to execute.

For example, the following **crontab** entry executes the **/home/bob/rpt.pl** script every weekday (Monday–Friday), every month, starting at 8:00 in the morning and every half hour until 16:30 in the afternoon (4:30 p.m.):

```
0,30 8-16 * 1-12 1-5 /home/bob/rpt.pl
```

As the administrator, you can use configuration files to determine whether a user can use the **crontab** command. The **/etc/cron.deny** and **/etc/cron.allow** files are used to control access to the **crontab** command. The format of each of these files is one username per line. Here's an example:

```
[root@OCS ~]$ cat /etc/cron.deny
alias
backup
bin
daemon
ftp
games
gnats
guest
irc
lp
mail
man
nobody
operator
proxy
sync
sys
www-data
```

Table 4.3 describes how the **/etc/cron.deny** and **/etc/cron.allow** files work.

TABLE 4.3 **How the /etc/cron.deny and /etc/cron.allow Files Work**

Situation	Description
Only the **/etc/cron.deny** file exists.	All users listed in this file are denied access to the **crontab** command, whereas all other users can execute the **crontab** command successfully. Use this file when you want to deny access to a few users but allow access to most users.
Only the **/etc/cron.allow** file exists.	All users listed in this file are allowed access to the **crontab** command, whereas all other users cannot execute the **crontab** command successfully. Use this file when you want to allow access to a few users but deny access to most users.
Neither file exists.	On most Linux distributions, this means that only the root user can use the **crontab** command. However, on some platforms, this results in all users being allowed to use the **crontab** command.
Both files exist.	Only the **/etc/cron.allow** file is consulted, and the **/etc/cron.deny** file is completely ignored.

The **/etc/crontab** file acts as the system **crontab**. The system administrator edits this file to enable the execution of system-critical processes at specific intervals. The following is a sample **/etc/crontab** file:

```
[root@OCS ~]$ cat /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin


#  m h dom mon dow user command
17 *      * * * root cd / && run-parts /etc/cron.hourly
```

Each configuration line describes a process to execute, when to execute it, and what username to execute the process as. Each line is broken into fields, separated by one or more space characters. Table 4.4 describes these fields.

TABLE 4.4 **Fields of a crontab Entry**

Field	Description
First field: **Minute**	The minute that the command should execute. Values can be 0–59. You can use a single value, a list of values (such as 0,15,30,45), or a range of values (such as 1–15). You can use the * character to indicate “all possible values.”
Second field: **Hour**	The hour that the command should execute. Values can be 0–23. You can use a single value, a list of values (such as 0,6,12,18), or a range of values (such as 8–16). You can use the * character to indicate “all possible values.”

Field	Description
Third field: **Day of the Month**	The day of the month that the command should execute. Values can be 1–31. You can use a single value, a list of values (such as 1,15), or a range of values (such as 1–10). You can use the * character to indicate "not specified" unless the fifth field is also an * character, in which case the * character means "all possible values."
Fourth field: **Month**	The month that the command should execute. Values can be 1–12. You can use a single value, a list of values (such as 6,12), or a range of values (such as 1–3). You can use the * character to indicate "all possible values."
Fifth field: **Day of the Week**	The day of the week that the command should execute. Values can be 0–7 (where 0=Sunday, 1=Monday,…6=Saturday, 7=Sunday). You can use a single value, a list of values (such as 1,3,5), or a range of values (such as 1–5). You can use the * character to indicate "not specified" unless the fifth field is also an * character, in which case the * character means "all possible values."
Sixth field: **Username**	The name of the user that the command should run as.
Seventh field: **Command Name**	The name of the command to execute.

ExamAlert

You will be expected to know the different fields of a **crontab** file for the Linux+ XK0-005 **exam.**

at

The **at** command is used to schedule one or more commands to be executed at one specific time in the future.

Syntax:

```
at time
```

where *time* indicates when you want to execute the command. For example, the following command allows you to schedule a command to run at 5 p.m. tomorrow:

```
at 5pm tomorrow
at>
```

When you see the **at>** prompt, you can enter a command to execute at the specified time. To execute multiple commands, press the Enter key to get another **at>** prompt. When this is complete, hold down the Ctrl key and press the d key. This results in an **<EOT>** message and creates the **at** job. Here's an example:

```
[root@OCS ~]$ at 5pm tomorrow
at> /home/bob/rpt.pl
at> echo "report complete" | mail bob
at> <EOT>
job 1 at Thu Feb 23 17:00:00 2017
```

Table 4.5 lists some important options for the **at** command.

TABLE 4.5 **at Command Options**

Option	Description
-m	Sends the user who created the **at** job an email when the job is executed.
-f *filename*	Reads commands from *filename*. This is useful when you are running the same **at** jobs on an infrequent basis.
-v	Displays the time and date when the **at** job will be executed.

The **atq** command lists the current user's **at** jobs:

```
[root@OCS ~]$ atq
1 Thu Feb 23 17:00:00 2017 a bob
```

The output includes a job number (**1** in this example), the date that the command will execute, and the user's name (**bob** in this example).

The **atq** command has no commonly used options.

To remove an **at** job before it is executed, run the **atrm** command followed by the job number to remove, as shown in this example:

```
[root@OCS ~]$ atq
1 Thu Feb 23 17:00:00 2017 a bob
[root@OCS ~]$ atrm 1
[root@OCS ~]$ atq
```

The **atrm** command has no commonly used options.

As the administrator, you can use configuration files to determine whether a user can use the command. The **/etc/at.deny** and **/etc/at.allow** files are used to control access to the **at** command.

The format of each of these files is one username per line. Here's an example:

```
[root@OCS ~]$ cat /etc/at.deny
alias
backup
bin
daemon
ftp
games
gnats
guest
irc
lp
mail
man
nobody
operator
proxy
sync
sys
www-data
```

Table 4.6 describes how the **/etc/at.deny** and **/etc/at.allow** files work.

TABLE 4.6 **How the /etc/at.deny and /etc/at.allow Files Work**

Situation	Description
Only the **/etc/at.deny** file exists.	All users listed in this file are denied access to the **at** command, and all other users can execute the **at** command successfully. Use this file when you want to deny access to a few users but allow access to most users.
Only the **/etc/at.allow** file exists.	All users listed in this file are allowed access to the **at** command, and all other users cannot execute the **at** command successfully. Use this file when you want to allow access to a few users but deny access to most users.

Situation	Description
Neither file exists.	On most Linux distributions, this means that only the root user can use the **at** command. However, on some platforms, this results in all users being allowed to use the **at** command.
Both files exist.	Only the **/etc/at.allow** file is consulted, and the **/etc/at.deny** file is completely ignored.

ExamAlert

Remember that **crontab** is for scheduling a process to run routinely, and **at** is used to schedule a process to run once.

Process Management

Process management includes listing running processes (that is, jobs or commands) and sending signals to the processes to have them modify their behavior (restart, stop, and so on). This section describes process management features.

Kill Signals

The **kill** command can be used to change the state of a process, including stopping (killing) it.

Syntax:

`kill` *`PID|jobnumber`*

To stop a process, first determine its process ID or job number and then provide that number as an argument to the **kill** command, as in this example:

```
[student@OCS ~]$ jobs
 [1]- Running                 sleep 999 &
 [2]+ Running sleep 777 &
 [student@OCS ~]$ kill %2
 [student@OCS ~]$ jobs
 [1]- Running                 sleep 999 &
```

```
[2]+ Terminated                 sleep 777
[student@OCS ~]$ ps -fe | grep sleep
student 17846 12540 0 14:30 pts/2 00:00:00 sleep 999
student 17853 12540 0 14:31 pts/2 00:00:00
grep --color=auto sleep
[student@OCS ~]$ kill 17846
[student@OCS ~]$ ps -fe | grep sleep
student 17856 12540 0 14:31 pts/2 00:00:00
grep --color=auto sleep
[1]+ Terminated                 sleep 999
```

Table 4.7 lists some important **kill** options.

TABLE 4.7 **kill Command Options**

Option	Description
-9	Forces a kill. Used when a process doesn't exit when a regular **kill** command is executed.
-l	Provides a list of other numeric values that can be used to send different kill signals to a process.

ExamAlert

For the Linux+ XK0-005 exam, keep in mind that kill -9 should only be used when all other attempts to stop a process have failed.

SIGTERM

A SIGTERM signal, also known as signal 15, is the default signal sent to a process when you use the **kill** command. This signal is a request for the process to stop, but the process can be programmed to ignore SIGTERM signals.

SIGKILL

A SIGKILL signal, also known as signal 9, is the signal sent to a process when you use the **kill** command with the **-9** option, as in this example:

```
kill -9 17846
```

The SIGKILL signal forces the process to stop without providing it the opportunity to shut down gracefully. If you use a SIGKILL signal on a process, you can lose data that the process had stored in memory.

SIGHUP

When a parent process is stopped, a hang-up (SIGHUP) signal is sent to all the child processes. This HUP signal is designed to stop the child processes. By default, a child process stops when sent an SIGHUP signal, but the process can be programmed to ignore SIGHUP signals. You can also have a process ignore a SIGHUP signal by executing the child process with the **nohup** command:

```
[student@OCS ~]$ nohup some_command
```

You typically use this technique when you remotely log in to a system and want to have some command continue to run even if you are disconnected. When you are disconnected, all of the programs you have running are sent HUP signals. Using the **nohup** command allows this specific process to continue running.

Listing Processes and Open Files

Several Linux commands can be used to list processes and the files that are opened by the processes. This section covers these commands.

top

The **top** command displays process information that is updated on a regular basis (by default, every 2 seconds). The first half of the output of the **top** command contains overall information, and the second half displays a select list of processes (by default, the processes that are using the CPU the most).

Figure 4.2 shows some typical output of the **top** command.

```
top - 16:09:10 up 2 days,  3:07,  2 users,  load average: 0.00, 0.07, 0.12
Tasks: 119 total,   2 running, 117 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.3 us,  1.0 sy,  0.0 ni, 97.0 id,  0.3 wa,  0.0 hi,  0.3 si,  0.0 st
KiB Mem:   4048292 total,  3832140 used,   216152 free,   356468 buffers
KiB Swap:        0 total,        0 used,        0 free.  1610568 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
26159 root      20   0 2461400 1.243g  24040 S  2.7 32.2  44:59.94 java
  965 root       0 -20       0      0      0 S  0.3  0.0   0:35.58 loop0
27545 nobody    20   0   87524   3616    892 S  0.3  0.1   0:05.32 nginx
28770 root      20   0   12824    940    776 S  0.3  0.0   0:14.39 ping
    1 root      20   0   33604   2952   1476 S  0.0  0.1   0:00.98 init
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:05.72 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   1:12.43 rcu_sched
    8 root      20   0       0      0      0 R  0.0  0.0   1:39.49 rcuos/0
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcuob/0
   11 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
```

FIGURE 4.2 **top Command Output**

Table 4.8 describes the output displayed in Figure 4.2.

TABLE 4.8 **top Command Output Walkthrough**

Output	Description
First line	Output derived from the **uptime** command.
Second line	A summary of processes running on the system.
Third line	CPU statistics since the last time **top** data was refreshed.
Fourth line	Physical memory statics. (Note: Type **E** while in the **top** command to change the value from kilobytes to another value.)
Fifth line	Virtual memory statics.
Remaining lines	A list of processes and associated information.

While the **top** command is running, you can use interactive commands to perform actions such as change display values, reorder the process list, and kill processes. These interactive commands are single characters. Table 4.9 describes the most important interactive commands.

TABLE 4.9 **Interactive Commands to Use with top**

Command	Description
h	Opens help. Displays a summary of interactive commands.
E	Changes the default value from kilobytes to another value; values "cycle" around back to kilobytes.
Z	Toggles color highlighting on; use lowercase **z** to toggle between color and non-color.

Command	Description
B	Toggles bold on and off.
< >	Moves the sort column to the left (<) or to the right (>).
s	Sets the update value to a different value than the default of 2 seconds.
k	Kills a process based on the process ID (PID).
q	Quits the **top** command.

The **top** command also supports several command-line options, including those listed in Table 4.10.

TABLE 4.10 **top Command-Line Options**

Option	Description
-d	Sets the time between data refresh.
-n *number*	Specifies the maximum number of data refreshes until the **top** command exits.
-u *username*	Displays only processes owned by *username*.

ps

The **ps** command is used to list processes that are running on the system. With no arguments, the command lists any child process of the current shell as well as the BASH shell, as shown here:

```
[student@OCS ~]$ ps
 PID TTY      TIME CMD
18360 pts/0   00:00:00 bash
18691 pts/0   00:00:00 ps
```

The **ps** command is unusual in that it supports older BSD options that normally don't have a hyphen (**-**) character in front of them.

Table 4.11 details some important options for the **ps** command.

TABLE 4.11 **ps Command Options**

Option	Description
-e	Displays all processes running on the system; the BSD method **ps ax** can also be used.
-f	Displays full information (that is, additional information about each process).

Option	Description
-u *username*	Displays all processes owned by *username*.
-forest	Provides a process hierarchy tree.

lsof

The **lsof** command is used to list open files. When used with no arguments, it lists all the open files for the OS, as shown here:

```
[root@OCS ~]# lsof | wc -l
25466
```

A more useful technique would be to list all files related to open network connections, as shown here:

```
[root@OCS ~]# lsof -i
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
avahi-dae 674 avahi 13u IPv4 15730 0t0
            UDP       *:mdns
avahi-dae 674 avahi 14u IPv4 15731 0t0
            UDP        *:49932
sshd 1411 root 3u IPv4 18771 0t0
            TCP       *:ssh (LISTEN)
sshd 1411 root 4u IPv6 18779 0t0
            TCP      *:ssh   (LISTEN)
master 2632 root 14u IPv4 20790 0t0
           TCP       localhost:smtp (LISTEN)
master 2632 root 15u IPv6 20791 0t0
            TCP      localhost:smtp (LISTEN)
dnsmasq 2739 nobody 3u IPv4 21518 0t0
            UDP      *:bootps
dnsmasq 2739 nobody 5u IPv4 21525 0t0
            UDP       192.168.122.1:domain
dnsmasq 2739 nobody 6u IPv4 21526 0t0
            TCP       192.168.122.1:domain (LISTEN)
cupsd 4099 root 12u IPv6 564510 0t0
            TCP       localhost:ipp (LISTEN)
```

```
cupsd 4099 root 13u IPv4 564511 0t0
            TCP        localhost:ipp (LISTEN)
dhclient 26133 root 6u IPv4 1151444 0t0
            UDP           *:bootpc
dhclient 26133 root 20u IPv4 1151433 0t0
               UDP       *:14638
dhclient 26133 root 21u IPv6 1151434 0t0
               UDP   *:47997
```

Table 4.12 describes common options for the **lsof** command.

TABLE 4.12 **lsof Command Options**

Option	Description
-i	Matches the Internet address; could also be used to display IP version (**-i4** or **-i6**) or port (**-i TCP:80**) or to display all open connections.
-u *user*	Lists files opened by *user*.
-p *pid*	Lists files opened by the process with the process ID *pid*.

htop

The **htop** command is much like the **top** command (see the "top" section, earlier in this chapter), but it has some additional features. For example, you can scroll through the output, both horizontally and vertically. The **htop** command also displays more information, such as user and kernel threads (where a *thread* is part of the execution of a process), which makes it a valuable command for software developers.

Setting Priorities

nice values are used to indicate to the CPU which process has the highest priority for access to the CPU. The values range from –20 (highest priority) to 19 (lowest priority). The default priority of any job created by a user is 0.

ExamAlert

nice values are often the subject of Linux+ XK0-005 exam questions. Remember that the lower the number, the higher the priority. Also remember that 0 is the default and that only the root user can run a process with a negative number. Finally, keep in mind that –20 is the lowest and 19 is the highest; you can expect to see confusing questions on the exam that provide numbers like –19 and 20 as possible answers.

The next section provides details on how to set a different priority when executing a command.

nice

To specify a different **nice** value than the default, execute the job via the **nice** command:

```
[student@OCS ~]$ nice -n 5 firefox
```

Note that a regular user cannot assign a negative **nice** value. These values can only be used by the root user. There are no additional useful options besides the **-n** option.

To view the **nice** value of a process, use the **-o** option with the **ps** command and include the value **nice**, as shown here:

```
[student@OCS ~] ps -o nice,pid,cmd
 NI PID CMD
 0 23865 -bash
 0 27969 ps -o nice,pid,cmd
```

renice

Use the **renice** command to change the **nice** value of an existing job. Here is an example:

```
[student@OCS ~] ps -o nice,pid,cmd
 NI PID CMD
 0 23865 -bash
 5 28235 sleep 999
 0 28261 ps -o nice,pid,cmd
[student@OCS ~] renice -n 10 -p 28235
28235 (process ID) old priority 5, new priority 10
```

```
[student@OCS ~] ps -o nice,pid,cmd
 NI PID CMD
 0 23865 -bash
 10 28235 sleep 999
 0 28261 ps -o nice,pid,cmd
```

Note

Regular (non-root) users can only change the priority of an existing process to a lower priority. Only the root user can alter a process priority to a higher priority.

Table 4.13 details some important options for the **renice** command.

TABLE 4.13 **renice Command Options**

Option	Description
-g *group*	Changes the priority of all files owned by *group*.
-u *user*	Changes the priority of all files owned by *user*.

Process States

Each process is assigned a state, depending on the current actions the process is taking (or if it is not taking any actions at all). This section describes the important process states.

Note that the **ps** and **top** commands can display the state a process is currently in. See the "ps" and "top" sections, earlier in this chapter, for more details.

Zombie

A *zombie process* is a process that has terminated but still has not been entirely cleared out of memory. Each process is started by another process, creating a "parent/child" relationship. When a child process ends, the parent process is responsible for telling the system that all details about the child process should be removed from memory.

In some rare cases, a child process may end without the parent being aware. This results in a zombie process. Zombie processes are fairly rare on modern Linux systems and typically indicate a bug that needs to be fixed.

Sleeping

A process that is in an uninterruptible sleep state is a process that is performing certain system calls that prevent it from being interrupted (that is, killed). Uninterruptible sleep state is fairly rarely seen on most modern Linux systems because these system calls are executed very quickly. If a process stays in uninterruptible sleep for a noticeable period of time, it is likely the result of a bug in the software.

A process that is in an interruptible sleep state is one that is performing some sort of I/O (input/output) operation, such as accessing a hard disk. This is a fairly common state, as I/O operations may take some time.

However, a process that is in interruptible sleep for a long period of time, especially if it is impacting the performance of the system, can indicate a problem. Either the device the process is attempting to access has an error (such as a bad data block on a hard disk) or the program has a bug.

Running

A running process is one that currently has operations taking place on the CPU or has operations on the CPU queue.

Stopped

A stopped process is no longer executing, but it might not have been cleared completely from memory.

Job Control

Job control is the ability to change the state of jobs. A job is a process that was started from the terminal window. This section describes commonly used commands for job control.

bg

A paused process can be restarted in the background by using the **bg** command:

```
[student@OCS ~]$ jobs
[1]+ Stopped                 sleep 999
[student@OCS ~]$ bg %1
[1]+ sleep 999 &
```

```
[student@OCS ~]$ jobs
[1]+ Running                 sleep 999 &
```

The **bg** command has no commonly used options.

> **Note**
>
> You can pause a process that is running in the foreground by holding down the Ctrl key and pressing z while in that process's window. See the "Ctrl+Z" section, later in this chapter, for more details.

fg

A paused process can be restarted in the foreground by using the **fg** command:

```
[student@OCS ~]$ jobs
[1]+ Stopped sleep 999
[student@OCS ~]$ fg %1
sleep 999
```

The **fg** command has no commonly used options.

> **Note**
>
> You can pause a process that is running in the foreground by holding down the Ctrl key and pressing z while in that process's window. See the "Ctrl+Z" section, later in this chapter, for more details.

jobs

The **jobs** command displays processes that are currently running and that were started from the shell in which you type the **jobs** command:

```
[student@OCS ~]$ jobs
[1]+ Stopped sleep 999
```

If you open another terminal window and type the **jobs** command, you don't see the processes that were displayed in your first terminal.

Ctrl+Z

When a process is running in the foreground, you can have a SIGTSTP signal sent to a process by holding down the Ctrl key and pressing the z key. A SIGTSTP signal is designed to pause a program. The program can then be restarted by using either the **bg** or **fg** command.

See the "**bg**" and "**fg**" sections, earlier in this chapter, for more details.

Ctrl+C

When a process is running in the foreground, you can have a SIGINT signal sent to a process by holding down the Ctrl key and pressing the c key. A SIGINT signal is designed to stop a program prematurely.

Ctrl+D

When you are running a process that accepts user input, such as the **at** command, you can send the process a signal that says "I am done providing input" by holding down the Ctrl key and pressing the d key (refer to the section "**at**," earlier in this chapter).

When you have finished entering **at** commands, hold down the Ctrl key and press the d key. This results in an **<EOT>** message and creates the **at** job. Here's an example:

```
[root@OCS ~]$ at 5pm tomorrow
at> /home/bob/rpt.pl
at> echo "report complete" | mail bob
at> <EOT>
job 1 at Thu Feb 23 17:00:00 2017
```

pgrep

Typically you use a combination of the **ps** and **grep** commands to display specific processes, like so:

```
[student@OCS ~]$ ps -e | grep sleep
25194 pts/0 00:00:00 sleep
```

However, the **pgrep** command can provide similar functionality:

```
[student@OCS ~]$ pgrep sleep
25194
```

Table 4.14 details some important options for the **pgrep** command.

TABLE 4.14 **pgrep Command Options**

Option	Description
-G *name*	Matches processes by group name.
-l	Displays the process name and PID.
-n	Displays the most recently started processes first.
-u *name*	Matches processes based on username.

pkill

When sending signals to a process using the **kill** command, you indicate which process by providing a process ID (PID). With the **pkill** command, you can provide a process name, a username, or another method to indicate which process or processes to send a signal to. For example, the following command sends a kill signal to all processes owned by the user sarah:

```
[student@OCS ~]$ pkill -u sarah
```

Table 4.15 details some important options for the **pkill** command.

TABLE 4.15 **pkill Command Options**

Option	Description
-G *name*	Matches processes by group name.
-u *name*	Matches processes by username.

pidof

The **pidof** command is useful when you know the name of a command that you want to control, but you don't know the command's PID (process ID). This command looks up a process by name and returns its PID:

```
[student@OCS ~]$ pidof dovecot
688
```

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. Which command can be used to control services?
   - ❍ **A. systemd**
   - ❍ **B. system**
   - ❍ **C. systemcfg**
   - ❍ **D. systemctl**

2. You run the **crontab -l** command and see the following output:

   ```
   15 12 * 3 1 /home/bob/rpt.pl
   ```

   When will this command execute?
   - ❍ **A.** At 3:15 on December 1
   - ❍ **B.** Every Monday of December at 3:15
   - ❍ **C.** Every Monday of March at 12:15
   - ❍ **D.** Never; this is an invalid **crontab** entry.

3. Which signal will be ignored when you run a process using the **nohup** command?
   - ❍ **A. -s**
   - ❍ **B. -h**
   - ❍ **C. -l**
   - ❍ **D.** None of these answers are correct.

4. Which of the following process priorities can a non-root user use?
   - ❍ **A. -20**
   - ❍ **B. -10**
   - ❍ **C. -1**
   - ❍ **D. 10**

Cram Quiz Answers

1. **D.** The **systemctl** command is used to control the state of services. The term **systemd** refers to the daemon that is running, not the command to control the services. The other answers are not valid commands for controlling services.

2. **C.** The time fields of the **crontab** output are in this order: Minute, Hour, Day of the Month, Month of the Year, and Day of the Week. Given this, the time the command would run is 12:15. No day of the month is specified (* means "not specified"), but **1** in the Day of the Week field means "every Monday," while the **3** value in the Month field means "March" (which is the third month of the year).
3. **A.** You can have a process ignore a SIGUP signal by executing the child process with the **nohup** command.
4. **D.** Non-root users cannot use negative number priorities.

This page intentionally left blank

CHAPTER 5

Use the Appropriate Networking Tools or Configuration Files

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.5:** Given a scenario, use the appropriate networking tools or configuration files.

The goal of this chapter is to provide you with an understanding of different networking tools and configuration files. You will first learn about a variety of tools that are designed to display network information and allow you to make dynamic changes to network settings. Next, you will learn about which configuration files to edit to make network changes that will be persistent across a reboot.

This chapter also explores tools that allow you to monitor network traffic and verify that the network is behaving correctly. Finally, you will learn about some tools that allow you to connect to remote systems, either to log in to these systems or to transfer data.

This chapter provides information on the following topics: interface management, name resolution, network monitoring, and remote networking tools.

Interface Management

An *interface* (or, more specifically, a *network interface*) in Linux is a device that provides access to the network. This section covers the tools that allow you to manage network interfaces.

iproute2 Tools

iproute2 tools refers to a collection of tools that are designed to replace the legacy net tools. (See the section "net-tools," later in this chapter,

for more details about that collection.) The iproute2 tools include the **ip** and **ss** commands, which are covered in this section.

ip

The **ip** command is a newer command that is designed to replace a collection of commands related to network interfaces.

Syntax:

```
ip [options] object command
```

Table 5.1 describes some of the most important **ip** command objects.

TABLE 5.1 **ip Command Objects**

Object	Refers to
addr	IPv4 or IPv6 address
link	Network device
route	Routing table entry

Table 5.2 describes some of the most important **ip**-related commands that can be executed.

TABLE 5.2 **Commands to Add, Delete, and Analyze Objects**

Command	Description
add	Adds an object.
delete	Deletes an object.
show (or **list**)	Displays information about an object.

The following example displays network information for devices, much like the **ifconfig** command:

```
[root@OCS ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
     inet 127.0.0.1/8 scope host lo
     valid_lft forever preferred_lft forever
```

```
     inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
     link/ether 08:00:27:b0:dd:dc brd ff:ff:ff:ff:ff:ff
     inet 192.168.1.26/24 brd 192.168.1.255 scope global dynamic
enp0s3
     valid_lft 2384sec preferred_lft 2384sec
     inet 192.168.1.24/16 brd 192.168.255.255 scope global enp0s3
     valid_lft forever preferred_lft forever
     inet6 fe80::a00:27ff:feb0:dddc/64 scope link
     valid_lft forever preferred_lft forever
```

ExamAlert

The **ip** command was designed to replace utilities like **ipconfig**, **arp**, and **route**. This may come up in questions on the Linux+ XK0-005 exam.

SS

The **ss** command is used to display socket information. Think of a socket as an existing network connection between two systems.

Syntax:

```
ss [options]
```

Without any options, this command lists all open sockets. For example:

```
[root@OCS ~]# ss | wc -l
160
[root@OCS ~]# ss | head
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port
u_str ESTAB 0 0 /var/run/dovecot/anvil 23454966 * 23454965
u_str ESTAB 0 0 /var/run/dovecot/anvil 23887673 * 23887672
u_str ESTAB 0 0 /run/systemd/journal/stdout 13569 * 13568
u_str ESTAB 0 0                             * 13893 * 13894
u_str ESTAB 0 0                             * 13854 * 13855
u_str ESTAB 0 0                             * 13850 * 13849
```

```
u_str ESTAB 0 0                              * 68924 * 68925
u_str ESTAB 0 0                              * 17996 * 17997
u_str ESTAB 0 0     /var/run/dovecot/config 9163531 * 9163871
```

Table 5.3 describes some useful options for the **ss** command.

TABLE 5.3 **ss Command Options**

Option	Description
-lt	Lists listening TCP sockets.
-lu	Lists listening UDP sockets.
-lp	Lists the process ID that owns each socket.
-n	Specifies not to resolve IP addresses to hostnames or port numbers to port names.
-a	Displays all information.
-s	Displays a summary.

ExamAlert

Network essentials like TCP and UDP are not specific exam topics, so they are not covered in this book. CompTIA considers the Linux+ XK0-005 exam to be a higher level exam than the A+ exam. In other words, there is an assumption that you already have knowledge of network essentials. Not having that knowledge can affect your ability to understand questions on the Linux+ exam. If you don't have an understanding of networking basics, consider researching this topic before taking the Linux+ exam.

NetworkManager

The NetworkManager software is used to manage network devices in Linux. The primary command-line configuration tool for NetworkManager is **nmcli**, which is covered in this section.

nmcli

The **nmcli** command is used to configure NetworkManager, which is designed to detect and configure network connections.

Syntax:

```
nmcli [options] object [command]
```

Example:

```
[root@OCS ~]# nmcli device status
DEVICE      TYPE       STATE      CONNECTION
virbr0      bridge     connected  virbr0
enp0s3      ethernet   connected   enp0s3
lo          loopback   unmanaged   --
virbr0-nic  tun        unmanaged   --
```

object is one of the keywords listed in Table 5.4.

TABLE 5.4 **Objects to the nmcli Command**

Keyword	Description
connection	Manages network connections.
device	Manages a specific device.
general	Gets NetworkManager status information.
networking	Enables or disables networking or displays the current status.
radio	Finds radio (wireless) networking information and configuration.

Common **nmcli** commands are described in Table 5.5.

TABLE 5.5 **Commands That Can Be Used as Arguments to the nmcli Command**

Command	Description
status	Displays the current setting.
on|off	Turns a setting on (or off).
up|down	Brings an interface up (or down).
add|delete	Adds a new device or deletes an existing one.

net-tools

The term *net-tools* refers to a collection of Linux commands that display or modify network information. For example, **ifconfig**, **route**, **arp**, and **netstat** (which are covered in this section) are all part of the net-tools collection.

ifconfig

One of the commonly used commands for displaying network information is the **ifconfig** command. When executed with no arguments, it lists active network devices, as shown in the following example.

```
[root@onecoursesource ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.16  netmask 255.255.255.0  broadcast
192.168.1.255
        inet6 fe80::a00:27ff:fe52:2878  prefixlen 64  scopeid
0x20<link>
        ether 08:00:27:52:28:78  txqueuelen 1000  (Ethernet)
        RX packets 20141  bytes 19608517 (18.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2973  bytes 222633 (217.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 3320  bytes 288264 (281.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3320  bytes 288264 (281.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The output shows network information for two devices: the primary Ethernet network card (eth0) and the local loopback address (lo). If the system had additional Ethernet network cards, they would be displayed as eth1, eth2, and so on. The purpose of the loopback address is to allow software to communicate with the local system using protocols and services that would normally require the communication to occur on a network. In most cases, there is not much for you to administer or troubleshoot in regard to the loopback address.

ifcfg

ifcfg is a script that is specifically designed to replace the **ifconfig** functions of adding, deleting, and disabling IP addresses.

hostname

The **hostname** command can display or change the system hostname, as shown here:

```
[root@onecoursesource ~]# hostname
onecoursesource
[root@onecoursesource ~]# hostname myhost
[root@myhost ~]# hostname
myhost
```

arp

The **arp** command is used to view the ARP table or make changes to it. When executed with no arguments, the **arp** command displays the ARP table, as shown here:

```
# arp
Address      HWtype HWaddress          Flags Mask Iface
192.168.1.11 ether  30:3a:64:44:a5:02 C          eth0
```

If a remote system has its network card replaced, it may be necessary to delete an entry from the ARP table. This can be accomplished by using the **-d** option to the **arp** command:

```
# arp -i eth0 -d 192.169.1.11
```

Once the address has been removed from the ARP table, there should be no need to add the new address manually. The next time the local system uses this IP address, it sends a broadcast request on the appropriate network to determine the new MAC address.

route

The **route** command can be used to display the routing table:

```
[root@OCS ~]# route
Kernel IP routing table
Destination Gateway     Genmask      Flags Metric Ref Use Iface
default     192.168.1.1  0.0.0.0     UG    100     0   0  enp0s3
```

```
192.168.0.0 0.0.0.0      255.255.0.0  U    100      0    0  enp0s3
192.168.1.0 0.0.0.0      255.255.0.0  U    100      0    0  enp0s3
```

This information can also be displayed with the **ip** command:

```
[root@OCS ~]# ip route show
default via 192.168.1.1 dev enp0s3 proto static metric 100
192.168.0.0/16 dev enp0s3 proto kernel scope link src 192.168.1.24
metric 100
192.168.1.0/24 dev enp0s3 proto kernel scope link src 192.168.1.26
metric 100
192.168.122.0/24 dev virbr0 proto kernel scope link src 192.168.122.1
```

The **route** command can also be used to modify the default router:

```
route add default gw 192.168.1.10
```

To add a new router, execute the following command:

```
route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.3.100
```

This command sends all network packets destined for the 192.168.3.0/24 network to the 192.168.3.100 router.

> **Note**
>
> **route** command changes are temporary and will only survive until the next time the system is booted. Permanent changes are made within your system's configuration files, which vary from one distribution to another.

/etc/sysconfig/network-scripts/

The directory **/etc/sysconfig/network-scripts/** is found on Red Hat–based distributions, such as Red Hat Enterprise Linux, CentOS, and Fedora. It contains a collection of files that are used to configure network devices, which you can see from the output in the following command.

```
[root@OCS ~]# ls /etc/sysconfig/network-scripts/
ifcfg-eth0      ifdown-Team        ifup-plusb
ifcfg-lo        ifdown-TeamPort    ifup-post
ifdown          ifdown-tunnel     ifup-ppp
```

```
ifdown-bnep     ifup              ifup-routes
ifdown-eth      ifup-aliases      ifup-sit
ifdown-ippp     ifup-bnep         ifup-Team
ifdown-ipv6     ifup-eth          ifup-TeamPort
ifdown-isdn     ifup-ippp         ifup-tunnel
ifdown-post      ifup-ipv6        ifup-wireless
ifdown-ppp       ifup-ipx         init.ipv6-global
ifdown-routes    ifup-isdn        network-functions
ifdown-sit       ifup-plip        network-functions-ipv6
```

In most cases, you can probably guess what a configuration file is used for based on its name. For example, the **ifup-wireless** file is used to configure wireless networks. Many of these files also have comments that are used to describe the purpose and use of the file.

The file most commonly edited is **ifcfg-***interface*, where *interface* is the name of the network interface. For example, **ifcfg-eth0** is used to configure the eth0 device, as shown here:

```
[root@OCS ~]# more /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.0.100
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
```

Table 5.6 lists some common **ifcfg-***interface* configuration settings.

TABLE 5.6 **ifcfg-*interface* Configuration Settings**

Setting	Description
DEVICE	The name of the interface.
BOOTPROTO	Set to **static** if you manually provide network information, such as IP address, netmask, and gateway. Set to **dhcp** to have these values dynamically assigned from a DHCP server.
ONBOOT	Normally set to **yes** to activate this device during the boot process.
IPADDR	The IP address that should be assigned to the device.
NETMASK	The device's netmask.
GATEWAY	The default router for this interface.

Name Resolution

Name resolution is the process of determining the IP address that corresponds to a given hostname. Reverse name resolution is the process of determining the hostname that corresponds to a given IP address. This section describes commands and features related to name resolution.

nsswitch

nsswitch refers to Name Service Switch (NSS), which is a tool for determining where the system will look for name resolution.

The NSS configuration file, **/etc/nsswitch.conf**, is used by applications to determine the sources from which to obtain name-service information and in what order. For example, for networking, this file contains the location of the name server resolver, the utility that provides hostname-to-IP-address translation.

```
[root@onesourcesource ~]#grep hosts /etc/nsswitch.conf
#hosts:     db files nisplus nis dns
hosts:      files dns
```

In this example, **files dns** means "look at the local **/etc/hosts** file first and then look at the DNS server if the required translation isn't in the local file."

Table 5.7 describes common hostname-to-IP-address translation utilities.

TABLE 5.7 **Hostname-to-IP-Address Translation Utilities**

Utility	Description
files	The local **/etc/hosts** file
dns	A DNS server
NIS	A Network Information Service server

/etc/resolv.conf

The **/etc/resolv.conf** file contains a list of the DNS servers for the system. A typical file looks as follows:

```
[root@OCS ~]# cat /etc/resolv.conf
search sample999.com
nameserver 192.168.1
```

If you are using a utility such as NetworkManager to configure your network settings or if you are using a DHCP client, then this file is normally populated by those utilities. For servers, this file is typically manually defined.

Table 5.8 describes common settings for the **/etc/resolv.conf** file.

TABLE 5.8 **Common Settings for the /etc/resolv.conf File**

Setting	Description
nameserver	The IP address of the DNS server. There can be up to three **nameserver** lines in the file.
domain	Used to specify the local domain, which allows for the use of short names for DNS queries.
search	A list of optional domains to perform DNS queries when using short names.

systemd

The **systemd** utility can be used to enable and disable network functionality. See the "System Services" section in Chapter 4, "Configure and Use the Appropriate Processes and Services," for more information.

hostnamectl

The **hostnamectl** command can be used to view and change host and system information. When it is used with no arguments, information about the system is displayed:

```
# hostnamectl
 Static hostname: student-VirtualBox
         Icon name: computer-vm
           Chassis: vm
    Machine ID: 7235c52cf8114b8188c985c05afe75c9
       Boot ID: e6ba643d8da44542a90a26c4466adca7
 Virtualization: oracle
       Operating System: Ubuntu 18.04.1 LTS
        Kernel: Linux 4.15.0-43-generic
          Architecture: x86-64
```

The **set-hostname** option allows you to specify one of two types of hostnames:

- **--static:** Changes are made in the **/etc/hostname** file and are persistent across reboots.

- **--transient:** Changes only apply to currently booted system. No changes are made to the **/etc/hostname** file.

There is also a feature (the **--pretty** option) that allows you to make a more flexible hostname that breaks standard network hostname rules. With **--pretty** and **--static**, changes are made to the **/etc/machine-info** file.

resolvectl

The **resolvectl** command can perform DNS lookups. For example, it can be used as follows to resolve the hostname of google.com:

```
[root@OCS ~]#  resolvectl query google.com
google.com: 142.250.189.238


-- Information acquired via protocol DNS in 611.6ms.
-- Data is authenticated: no
```

Bind-utils

Bind-utils is a software package that provides many of the commonly used commands for performing DNS queries. Examples of commands found in the Bind-utils package include **dig**, **host**, and **nslookup**. These commands are described in greater detail in this section.

The Bind-utils package is not normally installed by default on some Linux distributions, and you may need to install it.

dig

The **dig** command is useful for performing DNS queries on specific DNS servers. The use of the command is demonstrated here:

```
[root@OCS ~]# dig google.com


; <<>> DiG 9.9.4-RedHat-9.9.4-38.el7_3 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56840
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;google.com.         IN     A

;; ANSWER SECTION:
google.com. 268    IN     A 216.58.217.206

;; Query time: 36 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Sun Mar 05 17:01:08 PST 2017
;; MSG SIZE rcvd: 55
```

To query a specific DNS server rather than the default DNS servers for your host, use the following syntax:

```
dig @server host_to_look_up
```

Table 5.9 describes common options for the **dig** command.

TABLE 5.9 **dig Command Options**

Option	Description
-f *file*	Uses the contents of *file* to perform multiple lookups; the file should contain one hostname per line.
-4	Indicates to only perform IPv4 queries.
-6	Indicates to only perform IPv6 queries.
-x *address*	Performs a reverse lookup (and returns the hostname when provided an IP address).

nslookup

The **nslookup** command is designed to perform simple queries on DNS servers.

Syntax:

```
nslookup hostname
```

Example:

```
[root@OCS ~]# nslookup google.com
Server:        8.8.8.8
```

```
Address:      8.8.8.8#53

Non-authoritative answer:
Name:     google.com
Address: 216.58.219.238
```

While this command is often referred to as obsolete, it is still often used on modern distributions. The **nslookup** command is normally run without options.

host

The **host** command is normally used to perform simple hostname-to-IP-address translation operations (also called *DNS queries*). Here is an example:

```
[root@OCS ~]# host google.com
google.com has address 172.217.4.142
google.com has IPv6 address 2607:f8b0:4007:800::200e
google.com mail is handled by 30 alt2.aspmx.l.google.com.
google.com mail is handled by 50 alt4.aspmx.l.google.com.
google.com mail is handled by 20 alt1.aspmx.l.google.com.
google.com mail is handled by 10 aspmx.l.google.com.
google.com mail is handled by 40 alt3.aspmx.l.google.com.
```

Table 5.10 describes common options for the **host** command.

TABLE 5.10 **host Command Options**

Option	Description
-t	Specifies the type of query that you want to display; for example, **host -t ns google.com** displays Google's name servers.
-4	Indicates to only perform IPv4 queries.
-6	Indicates to only perform IPv6 queries.
-v	Enters verbose mode, with output like that of the **dig** command.

WHOIS

The **whois** command is useful for determining which company or person owns a domain. Often the output also contains information regarding how to contact this organization, although this information might be redacted for privacy reasons. Here is an example:

```
# whois onecoursesource.com | head
   Domain Name: ONECOURSESOURCE.COM
   Registry Domain ID: 116444640_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.tucows.com
   Registrar URL: http://www.tucows.com
   Updated Date: 2016-01-15T01:49:45Z
   Creation Date: 2004-04-07T19:45:31Z
   Registry Expiry Date: 2021-04-07T19:45:31Z
   Registrar: Tucows Domains Inc.
   Registrar IANA ID: 69
   Registrar Abuse Contact Email:
```

Network Monitoring

Network monitoring is the process of watching traffic on the network to determine if there are network traffic issues. This section describes many of the commonly used network monitoring tools.

tcpdump

When troubleshooting network issues or performing network security audits, it can be helpful to view the network traffic, including traffic that isn't related to the local machine. The **tcpdump** command is a packet sniffer that allows you to view local network traffic.

By default, the **tcpdump** command displays all network traffic to standard output until you terminate the command. This could result in a dizzying amount of data flying by on your screen. You can limit the output to a specific number of network packets by using the **-c** options, as in this example:

```
# tcpdump -c 5
tcpdump: verbose output suppressed, use -v or -vv for full
 protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535
bytes
11:32:59.630873 IP localhost.43066 > 192.168.1.1.domain:
 16227+ A? onecoursesource.com. (37)
```

```
11:32:59.631272 IP localhost.59247 > 192.168.1.1.domain:
 2117+ PTR? 1.1.168.192.in-addr.arpa. (42)
11:32:59.631387 IP localhost.43066 > 192.168.1.1.domain:
 19647+ AAAA? onecoursesource.com. (37)
11:32:59.647932 IP 192.168.1.1.domain > localhost.59247:
 2117 NXDomain* 0/1/0 (97)
11:32:59.717499 IP 192.168.1.1.domain > localhost.43066:
 16227 1/0/0 A 38.89.136.109 (53)
5 packets captured
5 packets received by filter
0 packets dropped by kernel
```

Wireshark/tshark

Wireshark is an amazing network sniffer that provides both GUI-based and TUI-based tools. To start the GUI tool, execute the **wireshark** command. The output should be similar to that shown in Figure 5.1.

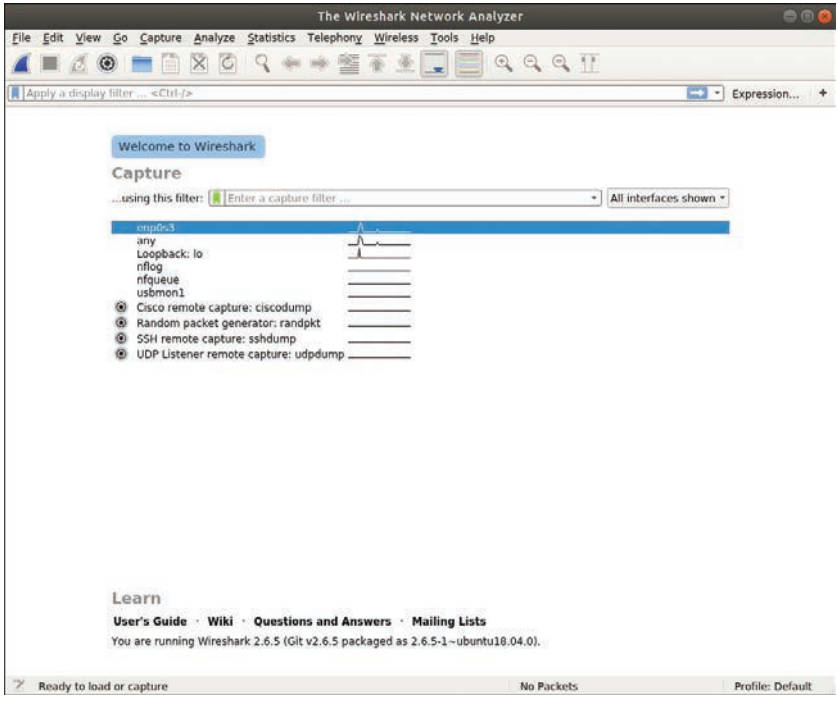

FIGURE 5.1 **The wireshark Command**

To view network traffic, you need to start a capture. Click **Capture**, **Start**. You can also limit what is captured by setting filters and options (click **Capture**, **Options**).

To use the TUI-based form of Wireshark, execute the **tstark** command as the root user. Here is an example:

```
# tshark
Capturing on 'enp0s3'
 1 0.000000000 10.0.2.15 → 68.105.28.11 DNS
 81 Standard query 0xeec4 A google.com OPT
 2 0.001031279 10.0.2.15 → 68.105.28.11 DNS
 81 Standard query 0x3469 AAAAgoogle.com OPT
 3 0.017196416 68.105.28.11 → 10.0.2.15 DNS
 109 Standard query response 0x3469 AAAA google.com AAAA
2607:f8b0:4007:800::200e OPT
 4 0.017265061 68.105.28.11 → 10.0.2.15 DNS
 97 Standard query response 0xeec4 A google.com A 172.217.14.110 OPT
 5 0.018482388 10.0.2.15 → 172.217.14.110 ICMP
 98 Echo (ping) request id=0x122c, seq=1/256, ttl=64
 6 0.036907577 172.217.14.110 → 10.0.2.15 ICMP
 98 Echo (ping) reply id=0x122c, seq=1/256, ttl=251 (request in 5)
 7 1.021052811 10.0.2.15 → 172.217.14.110 ICMP
 98 Echo (ping) request id=0x122c, seq=2/512, ttl=64
 8 1.039492225 172.217.14.110 → 10.0.2.15 ICMP
 98 Echo (ping) reply id=0x122c, seq=2/512, ttl=251 (request in 7)
```

netstat

The **netstat** command is useful for displaying a variety of network information. It is a key utility when troubleshooting network issues. Table 5.11 describes common options for the **netstat** command.

TABLE 5.11 **netstat Command Options**

Option	Description
-t or **--tcp**	Displays TCP information.
-u or **--udp**	Displays UDP information.
-r or **--route**	Displays the routing table.

Option	Description
-v or **--verbose**	Enters verbose mode and displays additional information.
-i or **--interfaces**	Displays information based on a specific interface.
-a or **--all**	Applies to all.
-s or **--statistics**	Displays statistics for the output.

For example, the following command displays all active TCP connections:

```
[root@OCS ~]# netstat -ta
Active Internet connections (servers and established)
Proto  Recv-Q  Send-Q Local Address          Foreign Address State
tcp    0       0       192.168.122.1:domain  0.0.0.0:* LISTEN
tcp    0       0       0.0.0.0:ssh           0.0.0.0:* LISTEN
tcp    0       0       localhost:ipp         0.0.0.0:* LISTEN
tcp    0       0       localhost:smtp        0.0.0.0:* LISTEN
tcp6   0       0       [::]:ssh              [::]:* LISTEN
tcp6   0       0       localhost:ipp          [::]:* LISTEN
tcp6   0      0      localhost:smtp         [::]:* LISTEN
```

traceroute

When you send a network packet to a remote system, especially across the Internet, it often needs to go through several gateways before it reaches its destination. You can see the gateways that the packet passes through by executing the **traceroute** command, as shown here:

```
# traceroute onecoursesource.com
traceroute to onecoursesource.com (38.89.136.109), 30 hops max, 60
byte packets
 1 10.0.2.2 (10.0.2.2) 0.606 ms 1.132 ms 1.087 ms
 2 b001649-3.jfk01.atlas.cogentco.com (38.104.71.201)
 0.738 ms 0.918 ms 0.838 ms
 3 154.24.42.205 (154.24.42.205) 0.952 ms 0.790 ms 0.906 ms
 4 be2629.ccr41.jfk02.atlas.cogentco.com (154.54.27.66)
 1.699 ms 1.643 ms 1.347 ms
 5 be2148.ccr41.dca01.atlas.cogentco.com (154.54.31.117)
 8.053 ms 7.719 ms 7.639 ms
 6 be2113.ccr42.atl01.atlas.cogentco.com (154.54.24.222)
```

```
 18.276 ms 18.418 ms 18.407 ms
 7 be2687.ccr21.iah01.atlas.cogentco.com (154.54.28.70)
 32.861 ms 32.917 ms 32.719 ms
 8 be2291.ccr21.sat01.atlas.cogentco.com (154.54.2.190)
 38.087 ms 38.025 ms 38.076 ms
 9 be2301.ccr21.elp01.atlas.cogentco.com (154.54.5.174)
 48.811 ms 48.952 ms 49.151 ms
10 be2254.ccr21.phx02.atlas.cogentco.com (154.54.7.33)
 57.332 ms 57.281 ms 56.896 ms
11 te2-1.mag02.phx02.atlas.cogentco.com (154.54.1.230)
 56.666 ms 65.279 ms 56.520 ms
12 154.24.18.26 (154.24.18.26) 57.924 ms 58.058 ms 58.032 ms
13 38.122.88.218 (38.122.88.218) 79.306 ms 57.740 ms 57.491 ms
14 onecoursesource.com (38.89.136.109) 58.112 57.884 ms 58.299 ms
```

ping

The **ping** command is used to verify that a remote host can respond to a network connection:

```
 [root@OCS ~]# ping -c 4 google.com
PING google.com (172.217.5.206) 56(84) bytes of data.
64 bytes from lax28s10-in-f14.1e100.net (172.217.5.206): icmp_seq=1
ttl=55 time=49.0 ms
64 bytes from lax28s10-in-f206.1e100.net (172.217.5.206): icmp_seq=2
ttl=55 time=30.2 ms
64 bytes from lax28s10-in-f14.1e100.net (172.217.5.206): icmp_seq=3
ttl=55 time=30.0 ms
64 bytes from lax28s10-in-f206.1e100.net (172.217.5.206): icmp_seq=4
ttl=55 time=29.5 ms


--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 29.595/34.726/49.027/8.261 ms
```

By default, the **ping** command continuously sends pings to the remote system until the user cancels the command (by pressing Ctrl+C). The **-c** option specifies how many ping requests to send.

mtr

If you want a really cool variation of the **traceroute** command (see the "**traceroute**" section, earlier in this chapter), install the **mtr** command. This command performs a **traceroute**-like operation every second, updating the display with statistics, as demonstrated in Figure 5.2.

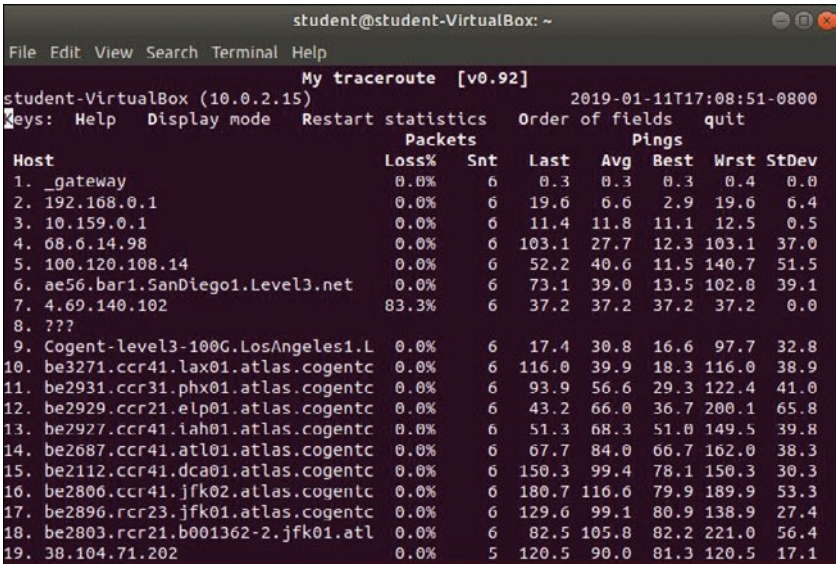

FIGURE 5.2 **The mtr Command**

Remote Networking Tools

Remote networking tools are designed to enable you to connect to remote systems for the following purposes:

- To execute commands on the remote systems
- To transfer files to the remote systems from your local system
- To transfer files from the remote systems to your local system

This section describes some of the most popular remote networking tools.

Secure Shell (SSH)

The **ssh** command is a utility that allows you to connect to a Secure Shell (SSH) server. The syntax of the command is as follows:

```
ssh user@hostname
```

where *user* is the username you want to use to log in as, and *hostname* is a system hostname or IP address.

The first time you use the **ssh** command to connect to a system, you see the following prompt:

```
[root@OCS ~]# ssh bob@server1
The authenticity of host 'server1' can't be established.
ECDSA key fingerprint is
 8a:d9:88:b0:e8:05:d6:2b:85:df:53:10:54:66:5f:0f.
Are you sure you want to continue connecting (yes/no)?
```

This ensures that you are logging in to the correct system. Typically users answer **yes** to this prompt, assuming that they are logging in to the correct machine, but this information can also be verified independently by contacting the system administrator of the remote system.

After the user answers **yes** to this prompt, the SSH server fingerprint is stored in the **known_hosts** file in the **~/.ssh** directory.

Table 5.12 describes common options for the **ssh** command.

TABLE 5.12 **ssh Command Options**

Option	Description
-F *configfile*	Specifies the configuration file to use for the **ssh** client utility. The default configuration file is **/etc/ssh/ssh_config**.
-4	Indicates to use only IPv4 addresses.
-6	Indicates to use only IPv6 addresses.
-E *logfile*	Places errors in the specified log file rather than displaying them to standard output.

SSH data for individual users is stored in each user's home directory under the **.ssh** subdirectory. This directory is used by SSH to store important data, and users can modify configurations in this directory. This section focuses on the files that may be stored in the **~/.ssh** directory.

After a connection is established with an SSH server, the SSH client stores the server's unique fingerprint key in the user's **.ssh/known_hosts** file, as shown here:

```
[root@OCS ~]# cat .ssh/known_hosts
|1|trm4BuvRf0HzJ6wusHssj6HcJKg=|EruYJY709DXorogeN5Hdcf6jTCo=
 ecdsa-sha2-nistp256AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzd
 HAyNTYAAABBBG3/rARemyZrhIuirJtfpfPjUVnph9S1w2NPfEWec/f59V7nA
 ztn5rbcGynNYOdnozdGNNizYAiZ2VEhJ3Y3JcE=
```

Typically the contents of this file should be left undisturbed; however, if the SSH server is reinstalled, it has a new fingerprint key. All users must then remove the entry for the SSH server in the **.ssh/known_hosts** file.

When a user wants to use password-based SSH authentication, the first step is to create authentication keys by using the **ssh-keygen** command.

After the authentication key files have been created, the public key (the contents of either the **~/.ssh/id_dsa.pub** or **~/.ssh/id_rsa.pub** file) needs to be copied to the system that the user is attempting to log in to. This requires placing the public key into the **~/.ssh/authorized_keys** file on the SSH server. This can be accomplished by manually copying over the content of the public key file from the client system and pasting it into the **~/.ssh/authorized_keys** file on the SSH server. Alternatively, you can use the **ssh-copy-id** command, which has the following syntax:

```
ssh-copy-id user@server
```

Users can customize how commands like **ssh**, **scp**, and **sftp** work by creating the **~/.ssh/config** file. The format and settings in this file are the same as those in the **/etc/sshd/ssh.conf** file.

cURL

The **curl** command allows for noninteractive data transfer from a large number of protocols, including the following:

- FTP
- FTPS
- HTTP
- SCP
- SFTP

- SMB
- SMBS
- Telnet
- TFTP

ExamAlert

You should consider memorizing the list of protocols that **curl** supports.

Although the **curl** command supports more protocols than the **wget** command, the **wget** command can perform recursive downloads and can recover from failed download attempts, so it is advantageous in certain situations. The **curl** command also supports wildcard characters. The goal of both of the commands is essentially the same.

Syntax:

```
curl location
```

Example:

```
# curl http://onecoursesource.com
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved
 <a href="http://www.onecoursesource.com/">here</a>.</p>
</body></html>
```

Note that the data is displayed, not stored in a file, as it is with the **wget** command. You can use redirection to put the contents into a file.

wget

The **wget** command is designed to be a noninteractive tool for downloading files from remote systems via HTTP, HTTPS, or FTP. It is often used within scripts.

Syntax:

```
wget location
```

Example:

```
# wget http://onecoursesource.com
--2019-01-09 15:18:26-- http://onecoursesource.com/
Resolving onecoursesource.com (onecoursesource.com)...38.89.136.109
Connecting to onecoursesource.com (onecoursesource.com)
 |38.89.136.109|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.onecoursesource.com/ [following]
--2019-01-09 15:18:26-- http://www.onecoursesource.com/
Resolving www.onecoursesource.com
 (www.onecoursesource.com)... 38.89.136.109
Reusing existing connection to onecoursesource.com:80.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

index.html [ <=> ]
 12.25K --.-KB/s in 0s

2019-01-09 15:18:26 (259 MB/s) - 'index.html' saved [12539]
```

Table 5.13 lists some useful options to use with the **wget** command.

TABLE 5.13 **wget Command Options**

Option	Description
-h	Displays help.
-b	Performs downloads in the background. This is useful for large downloads.
-q	Downloads quietly.
-v	Enters verbose mode.
-nc	Indicates not to clobber existing files.
-c	Continues a partial download. This is useful if a download fails due to a disconnect.
-r	Enters recursive mode.

nc

See the "**nc**" section in Chapter 2, "Manage Files and Directories."

rsync

See the "**rsync**" section in Chapter 2.

Secure Copy Protocol (SCP)

See the "**scp**" section in Chapter 2.

SSH File Transfer Protocol (SFTP)

The **sftp** command uses the SSH (Secure Shell) protocol to securely transfer files across the network. To access a remote system, use the following syntax:

```
sftp user@machine
```

After logging in to the remote system, you are provided with an interface that begins with the following prompt:

```
sftp>
```

At this prompt, several commands can be used to transfer files and perform other operations. Table 5.14 describes the most important of these commands.

TABLE 5.14 **Commands That Can Be Used at the sftp Prompt**

Command	Description
pwd	Displays the current remote directory.
lpwd	Displays the current local directory.
cd	Changes to a different directory on a remote system.
lcd	Changes to a different directory on the local system.
get	Downloads a file from the current directory on the remote system to the current directory on the local system. Use **-r** to get an entire directory structure.
put	Uploads a file from the current directory on the local system to the current directory on the remote system. Use **-r** to put an entire directory structure.
ls	Displays files in the current directory of a remote system.
lls	Displays files in the current directory of the local system.
exit	Quits **sftp**.

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. Which command enables you to configure NetworkManager?
   - ❍ **A. ifconfig**
   - ❍ **B. netconfig**
   - ❍ **C. nmcli**
   - ❍ **D. nmconfig**

2. Which of the following tools can display network route information? (Choose two.)
   - ❍ **A. route**
   - ❍ **B. ifconfig**
   - ❍ **C. arp**
   - ❍ **D. ip**

3. Which setting in the **/etc/sysconfig/network-scripts/ifcfg-eth0** file is used to set the default router?
   - ❍ **A. DEVICE**
   - ❍ **B. ROUTE**
   - ❍ **C. NETMASK**
   - ❍ **D. GATEWAY**

4. Which of the following commands allow you to display information about network packets? (Choose two.)
   - ❍ **A. tcpdump**
   - ❍ **B. wireshark**
   - ❍ **C. netstat**
   - ❍ **D. mtr**

Cram Quiz Answers

1. **C.** The **nmcli** command is used to configure NetworkManager, which is designed to detect and configure network connections.
2. **A and D.** The **route** command can be used to display the routing table. This information can also be displayed with the **ip** command, as follows: **ip route show**.
3. **D.** The **GATEWAY** setting is used to define the default router for an interface.
4. **A and B.** Both **tcpdump** and Wireshark capture network packet information, which you can then display. The **netstat** command provides overall network statistics, and **mtr** displays the hops to get from one system to another.

CHAPTER 6

Build and Install Software

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.6:** Given a scenario, build and install software.

This chapter explores how to manage software on different Linux distributions. You will learn the essentials of different Linux sandbox applications as well as the differences between kernel updates and system package updates.

This chapter provides information on the following topics: package management, sandboxed applications, and system updates.

Package Management

A *package* in Linux is a file that contains software that you can install on the system. The process of managing a package includes performing any of the following operations:

- Listing installed packages
- Viewing the contents of a package
- Installing a package
- Removing a package

This section reviews the various tools that are used to manage software packages on different platforms.

ExamAlert

While you might only work on one distribution, you need to be prepared for Linux+ XK0-005 exam questions related to package management on different distributions.

DNF

The DNF tool is designed as an enhancement and replacement for **yum**. (See the next section, "YUM," for more information.) The majority of the changes were made to the back end of the software. Most **dnf** commands work just like **yum** commands.

On the back end, the DNF tool handles dependencies better and addresses some additional YUM deficiencies (such as using older versions of Python). YUM hasn't been completely replaced, so knowing that either command may be used is important.

It is important to note that the configuration file for DNF is different from the configuration file for YUM. Configure DNF by editing the **/etc/dnf/dnf.conf** file.

YUM

The **yum** command is used to install software from repositories. It can also be used to remove software and display information regarding software. Table 6.1 highlights the primary **yum** commands and options.

TABLE 6.1 **yum Commands and Options**

Command/Option	Description
install	Installs a package and any dependency packages from a repository. Example: **yum install zip**.
groupinstall	Installs an entire software group from a repository. Example: **yum groupinstall "Office Suite and Productivity"**.
update	Updates the specified software package.
remove	Removes the specified software package and any dependency packages from the system.
groupremove	Removes the specified software group from the system.
list	Lists information about packages, including which packages are installed and which packages are available. Example: **yum list available**.
grouplist	Lists information about software groups, including what packages are part of a group; use **yum grouplist** with no arguments to see a list of available software groups.
info	Provides information about a specific software package. Example: **yum info zip**.
groupinfo	Provides information about a specific software group.

Command/Option	Description
-y	Answers **yes** automatically to any prompts. Example: **yum -y install zip**.

Table 6.2 describes some important options to the **yum list** command.

TABLE 6.2 **yum list Command Options**

Option	Description
all	Lists all packages that are installed or available.
installed	Lists all packages that are currently installed.
available	Lists all packages that are currently not installed but available for installation.
updates	Lists all packages that are currently installed on the system and that also have an available newer version on a repository.

Note

Wildcards (or *globs*) may be used with **yum** commands. Here's an example:

```
# yum list installed "*zip*"
Loaded plugins: fastestmirror, langpacks
Repodata is over 2 weeks old. Install yum-cron?
 Or run: yum makecache fast
Loading mirror speeds from cached hostfile
 * base: mirror.supremebytes.com
 * epel: mirror.chpc.utah.edu
 * extras: mirrors.cat.pdx.edu
 * updates: centos.sonn.com
Installed Packages
bzip2.x86_64                    1.0.6-13.el7   @base
bzip2-libs.x86_64               1.0.6-13.el7   @base
gzip.x86_64                     1.5-8.el7      @base
perl-Compress-Raw-Bzip2.x86_64 2.061-3.el7   anaconda
unzip.x86_64                    6.0-15.el7     @base
zip.x86_64                      3.0-10.el7     @anaconda
```

The **yumdownloader** command is used to download software packages without installing the software. The resulting RPM file could be installed manually or copied to other systems.

Table 6.3 describes some important options to the **yumdownloader** command.

TABLE 6.3 **yumdownloader Command Options**

Option	Description
--destdir	Used to specify the directory for downloading RPM files. (The default is the current directory.)
--resolve	Used to download dependency packages for the specified package. (The default is to download only specified packages.)
--source	Used to download the source RPM, not the binary (installable) RPM.

The **/etc/yum.conf** file is the primary configuration file for **yum** commands.

Example:

```
[main]
cachedir=/var/cache/yum/$basearch/$releasever
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=5
bugtracker_url=http://bugs.centos.org/set_project.php
 ?project_id=23&ref=http://bugs.centos.org/
 bug_report_page.php?category=yum
distroverpkg=centos-release
```

Table 6.4 describes some key settings of the **/etc/yum.conf** file.

TABLE 6.4 **/etc/yum.conf File Key Settings**

Setting	Description
cachedir	Directory where RPMs will be placed after download.
logfile	Location of the log file that contains **yum** actions.

Setting	Description
gpgcheck	A value of **1** means perform a GPG (GNU Privacy Guard) check to ensure that the package is valid; **0** means do not perform a GPG check. (This can be overridden by specific settings for each repository configuration file.)
assumeyes	A value of **1** means always assume "yes" to yes/no prompts; **0** means do not make any assumption (but provide a prompt instead).

ExamAlert

Expect a Linux+ XK0-005 exam question on the key settings described in Table 6.4.

The **/etc/yum.repos.d** directory contains files that end in **.repo** and that are used to specify the location of **yum** repositories. Each file defines one or more repositories, as illustrated in Figure 6.1.

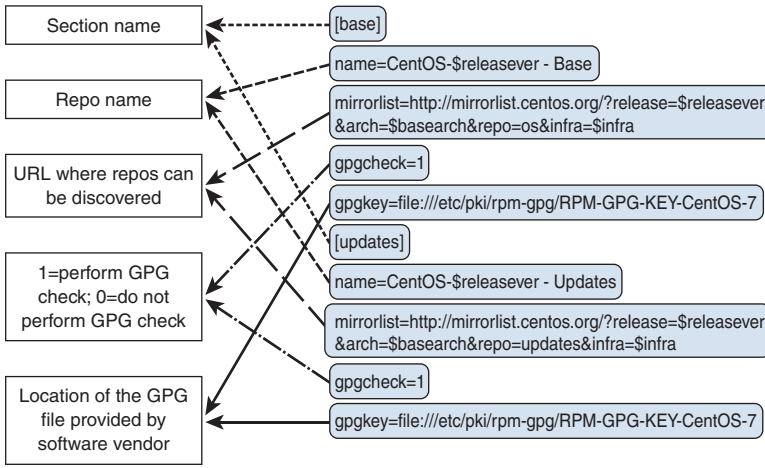

FIGURE 6.1 **The Format of Files in the /etc/yum.repos.d Directory**

APT

Use the **apt-get** command to manage Debian packages that are located on a repository. This command makes use of the **/etc/apt/sources.list** file to determine which repository to use. (See more about this file at the end of this section).

Here are some syntax examples of the **apt-get** command:

```
apt-get [options] command
apt-get [options] install|remove pkg1 [pkg2...]
apt-get [options] source pkg1 [pkg2...]
```

To specify what action to take, provide a keyword (command) to the **apt-get** command. Table 6.5 describes some useful commands for **apt-get**.

TABLE 6.5 **apt-get Commands**

Command	Description
install	Installs the specified package; if the package is currently installed, use the **--only-upgrade** option to upgrade rather than install fresh.
update	Updates the package cache of every available package.
upgrade	Updates all packages and their dependencies.
remove	Removes a package but leaves its configuration files on the system.
purge	Removes a package, including its configuration files.

Use the **apt-cache** command to display package information regarding the package cache.

Here are some syntax examples of the **apt-cache** command:

```
apt-cache [options] command
apt-cache [options] show pkg1 [pkg2...]
```

Example:

```
# apt-cache search xzip
xzip - Interpreter of Infocom-format story-files
```

To specify what action to take, provide a keyword (command) to the **apt-cache** command. Table 6.6 describes some useful commands for **apt-cache**.

TABLE 6.6 **apt-cache Commands**

Command	Description
search	Displays all packages with the search term listed in the package name or description; the search term can be a regular expression.
showpkg	Displays information about a package; the package name is provided as an argument.

Command	Description
stats	Displays statistics about the package cache (for example, **apt-cache stats**).
showsrc	Displays information about a source package; the package name is provided as an argument.
depends	Displays a package's dependencies.
rdepends	Displays a package's reverse dependencies (that is, packages that rely on this package).

The **aptitude** utility is a menu-driven tool designed to make it easy to display, add, and remove packages. Figure 6.2 shows this tool.

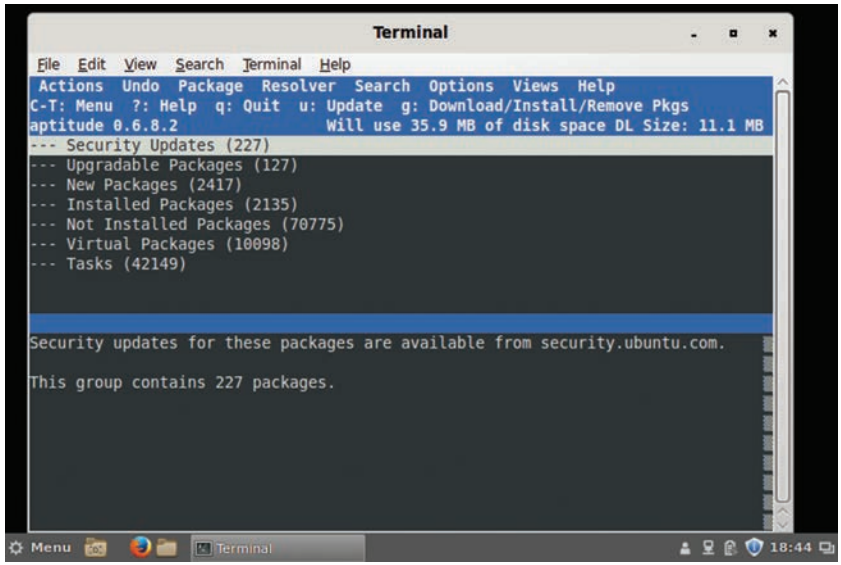

FIGURE 6.2 **The aptitude Utility Screen**

The **/etc/apt/sources.list** file contains a list of URLs of software repositories; there can also be files in the **/etc/apt/sources.list.d** directory that provide this information. Figure 6.3 describes this file.

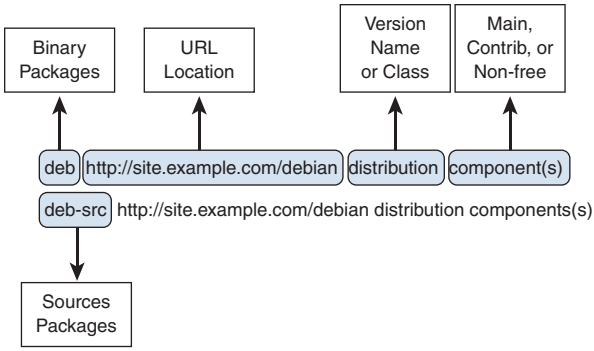

FIGURE 6.3 **The /etc/apt/sources.list File**

The distribution can be one of the following:

- Release name (for example, wheezy, jessie, stretch, or sid)
- Class name (for example, oldstable, stable, testing, or unstable)

The component can be one of the following:

- **main:** Packages must comply with DFSG (Debian Free Software Guidelines).
- **contrib:** Packages must comply with DFSG, but package dependencies do not have to.
- **non-free:** Packages do not comply with DFSG.

ExamAlert

Know the different components described in the previous list before you take the Linux+ XK0-005 exam.

The following is an example of the default file for the Jessie version of Debian:

```
deb http://httpredir.debian.org/debian jessie main
deb-src http://httpredir.debian.org/debian jessie main

deb http://httpredir.debian.org/debian jessie-updates main
deb-src http://httpredir.debian.org/debian jessie-updates main

deb http://security.debian.org/ jessie/updates main
deb-src http://security.debian.org/ jessie/updates main
```

The primary configuration file for APT is the **/etc/apt.conf** file. This file can be used to set debug options, configure proxy connections, and configure how APT caches packages.

RPM

The **rpm** command is useful for installing, upgrading, and removing packages that are already downloaded on your system. Table 6.7 describes some of the useful options.

TABLE 6.7 **rpm Command Options**

Option	Description
-i	Installs a package.
-U	Updates a package if an older version of the package exists; installs from scratch if the older version does not exist.
-F	Updates a package if an older version of the package exists; does nothing if an older version does not exist.
-e	Removes the package, including the configuration files.
-l	Lists packages that are currently installed.
-q	Performs a package query; additional options can be used to fine-tune the query.
-f	Determines which package a specific file belongs to.

Use the **-q** option to the **rpm** command to perform queries. Table 6.8 describes some additional options for fine-tuning a query.

TABLE 6.8 **rpm Query Options**

Option	Description
-a	Returns a list of all installed packages.
-c	Lists the configuration files installed with the specified package.
-d	Lists the documentation files installed with the specified package.
-i	Displays information about the specified package.
-K	Verifies the integrity of the specified package.
-l	Lists all files installed with the specified package.
-provides	Lists which capabilities the specified package provides.
-R	Lists which capabilities the specified package requires.
-s	Displays the state of each file that was installed by the specified package (normal, not installed, or replaced).

Example:

```
$ rpm -qc cups
/etc/cups/classes.conf
/etc/cups/client.conf
/etc/cups/cups-files.conf
/etc/cups/cupsd.conf
/etc/cups/lpoptions
/etc/cups/printers.conf
/etc/cups/snmp.conf
/etc/cups/subscriptions.conf
/etc/dbus-1/system.d/cups.conf
/etc/logrotate.d/cups
/etc/pam.d/cups
```

dpkg

Use the **dpkg** command to manage local Debian packages.

Syntax:

```
dpkg [option] command
```

Table 6.9 describes some useful **dpkg** options.

TABLE 6.9 **dpkg Command Options**

Option	Description
-i	Installs a package.
-r	Removes the package but keeps the configuration files.
-P	Removes the package, including the configuration files (purge).
-l	Lists the packages that are currently installed.
-L	Lists files that were installed with a package (for example, **dpkg -L zip**).
-V	Verifies the integrity of the specified package or packages.
-s	Displays package status.
-C	Checks for broken packages.
-S	Lists the name of the package that was responsible for a specific file being installed on the system (for example, **dpkg -S /usr/bin/zip**).

When a package is installed, it might run a configuration script as part of the installation process. To run this configuration script again at some point in the future, use the **dpkg-reconfigure** command. Although there are some options to this command, they are rarely used.

The syntax of the **dpkg-reconfigure** command is as follows:

```
dpkg-reconfigure [options] source packages
```

The following example reruns the **tzdata** configuration scripts:

```
dpkg-reconfigure tzdata
```

ZYpp

The ZYpp software package provides the zypper utility. The **zypper** utility is found on SUSE Linux. It is derived from the RPM software suite and works very similarly to **yum**. It has automatic dependency checking and uses repositories. Its command structure and options are almost identical to those of **yum**. Here's an example:

```
zypper install pkg_name
```

If you know how to use the **yum** command, then in most cases you can replace **yum** with **zypper**, and the command will work successfully.

Sandboxed Applications

One of the challenges in deploying applications on Linux is the many different distributions. It is very difficult for developers to properly test how well applications perform in all of the possible environments.

A sandbox can help solve this problem. A sandbox environment behaves the same, regardless of which Linux distribution it is placed on. A sandboxed application is any application that is installed within a sandbox environment.

ExamAlert

This topic covers some of the basics of popular Linux sandboxes. Keep in mind that the Linux+ XK0-005 exam won't ask you detailed questions about these environments.

snapd

The **snapd** daemon manages packages called *snaps*, which are downloaded from the Snap store maintained by Canonical (which also maintains Ubuntu Linux). In fact, Ubuntu Core is a Linux distribution that uses only snap applications.

Developers can use Snapcraft, a tool that allows developers to package their programs as snaps.

Flatpak

To use Flatpak, you first need to install the Flatpak package. This package is not available for all distributions, but most distributions have it available. Flatpak applications are installed from repositories that are called *remotes*.

The Flatpak applications are compiled for the Flatpak sandbox environment. These compiled applications typically include the required libraries, but libraries can also be shared between applications.

Flatpak applications are normally graphical applications designed for regular users. Note that Flatpak software can be larger than traditional RPM or DEP packages.

AppImage

AppImage uses a unique approach in that each AppImage package is fully self-contained. This means a single package file contains all of the software, including libraries. AppImage packages tend to be smaller than snapd or Flatpak packages.

System Updates

Packages can be grouped into two major categories: system packages or non-system packages. Think of system packages as any software than can affect the way the core operating system functions. Examples of system packages include:

- **iptables:** Software that provides firewall capabilities
- **libcrypt:** Software that provides encryption libraries
- **kernel:** Software that provides the kernel code

Non-system packages are packages that don't affect the core operating system functions. Think of applications that regular users will use but that aren't required for the operating system to function, including:

- Web browsers
- Email client programs
- Games

Updating non-system packages typically only requires the user to reset an application to start using the new version. Updating system packages might require more action by the system administrator. These system packages can also be broken into two categories: kernel and all others. Kernel updates are described in the "Kernel Updates" section that follows. All other system packages are described in the "Package Updates" section that concludes the chapter.

Kernel Updates

Kernel updates are different from other system updates in two ways:

- To use a new kernel, you need to reboot the operating system.
- When a kernel is installed, it doesn't replace the previous kernel. When a non-kernel software package is installed to upgrade a previous version, it replaces the original. Kernel packages are different because if the new kernel fails to boot the system, you want to be able to revert to the previous version.

Package Updates

One important aspect of system updates is that you might need to restart a service in order to have the updates take effect. This could be accomplished by a system reboot, but this technique is rarely needed as just restarting the service should be enough. In many cases, software packages have instructions built in to restart the software; however, as a system administrator, you are responsible for verifying that this has taken place.

Package updating is an important task that shouldn't be overlooked as many updates include security fixes. Failing to restart the software leaves your system vulnerable to attacks and other security issues.

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. The _____ command is used to download software packages without installing the software.
   - ❍ **A. rpm**
   - ❍ **B. yumdownloader**
   - ❍ **C. dpkg**
   - ❍ **D.** None of these answers are correct.

2. Which of the following is the primary configuration file for the **yum** command?
   - ❍ **A. /etc/yum-config/yum.conf**
   - ❍ **B. /etc/yum**
   - ❍ **C. /etc/yum.conf**
   - ❍ **D. /etc/yum.config**

3. Which option to the **rpm** command updates a package if an older version of the package exists but does nothing if an older version does not exist?
   - ❍ **A. -i**
   - ❍ **B. -U**
   - ❍ **C. -e**
   - ❍ **D. -F**

4. You have a kernel package installed on your system, and you install a new one by using the **rpm -i** command. How many kernel packages should you end up with after running this command?
   - ❍ **A.** 1
   - ❍ **B.** 2
   - ❍ **C.** 3
   - ❍ **D.** None as this results in kernel corruption.

Cram Quiz Answers

1. **B.** The **yumdownloader** command is used to download software packages without installing the software. The **rpm** and **dpkg** commands don't download packages; you must do that manually.
2. **C.** The **/etc/yum.conf** file is the primary configuration file for **yum** commands. The other answers are not valid files.

3. **D.** The **-F** option updates a package if an older version of the package exists but does nothing if an older version does not exist. The **-i** option is used to install (not update) a package. The **-U** option updates a package if an older version of the package exists and installs from scratch if the older version does not exist. The **-e** option deletes a package.
4. **B.** When installing a kernel package with the **rpm -i** command, the old kernel is left on the system, and the new kernel is installed in a different, non-conflicting location. As a result, you should have two kernels: the old one and the new one.

This page intentionally left blank

CHAPTER 7

Manage Software Configurations

This chapter covers the following Linux+ XK0-005 exam objective:

- **1.7:** Given a scenario, manage software configurations.

In Chapter 6, "Build and Install Software," you learned about managing software. In this chapter the focus shifts toward software configuration. You will learn how to manage software repository configurations and how to configure the kernel. You will also learn how to configure common system services, such as SSH, NTP, and syslog.

This chapter provides information on the following topics: updating configuration files, configuring kernel options, configuring common system services, and localization.

Updating Configuration Files

This section explores the package management configuration files that you should know how to update.

Procedures

If you upgrade a package, you might need to restart or reload a service. Typically the documentation that comes with the software package should indicate if a restart or reload is required. You also might need to restart or reload a service if changes are made to the configuration file for the software. This section addresses these procedures.

Restart Service

The **restart** option is used with the **systemctl** command to restart a service that is currently running. A restart can make the service unavailable for a short period of time.

Syntax:

```
systemctl restart process_name
```

You can determine whether a service is currently running by using the **active** command:

```
# systemctl active cups
enabled
```

Reload Service

The **reload** option is used with the **systemctl** command to reload a service that is currently running. This command does not stop the service but rather has the service reread its configuration file and change its behavior based on changes in the configuration file.

Syntax:

```
systemctl reload process_name
```

You can determine whether a service is currently running by using the **active** command:

```
# systemctl active cups
enabled
```

.rpmnew

When RPMs are installed, they might overwrite the default configuration files of the software package as new configuration settings are added or older configuration settings are removed.

The overwriting of the configuration files can lead to frustration if the system administrator has spent time and effort creating a customized configuration file. As a result, software developers might choose to place the new configuration file contents in a file called **.rpmnew**. Then a system administrator can

review the **.rpmnew** file and determine which new settings should be incorporated into the primary configuration file.

Another technique is for the software developers to use a **.rpmsave** file, as described in the next section.

.rpmsave

When RPMs are installed, they might overwrite the default configuration files of the software package as new configuration settings are added or older configuration settings are removed.

The overwriting of the configuration files can lead to frustration if the system administrator has spent time and effort creating a customized configuration file. As a result, software developers might choose to place the previous configuration file contents in a file called **.rpmsave**. Then a system administrator can review the **.rpmsave** file and determine which of the previous settings should be incorporated into the primary configuration file.

Another technique is for the software developers to use a **.rpmnew** file, as described in the previous section.

ExamAlert

Be ready to be tested on the difference between the **.rpmnew** and **.rpmsave** files.

Repository Configuration Files

Repository configuration files, discussed in Chapter 6, are an integral component of package management. The Linux+ XK0-005 exam lists these topics in different exam categories, so the following section headings have been preserved to inform you where to go to find details about these topics.

/etc/apt.conf

See the "APT" section in Chapter 6.

/etc/yum.conf

See the "YUM" section in Chapter 6.

/etc/dnf/dnf.conf

See the "DNF" section in Chapter 6.

/etc/yum.repo.d

See the "YUM" section in Chapter 6.

/etc/apt/sources.list.d

See the "APT" section in Chapter 6.

Configure Kernel Options

The Linux kernel is highly configurable. You can make changes to the kernel by using parameters and kernel modules. This section covers these configuration features.

Parameters

A kernel parameter is a value that changes the behavior of the kernel.

sysctl

You can view and change parameters by using the **sysctl** command. For example, to view all the kernel and kernel module parameters, execute the **sysctl** command with the **-a** option:

```
[root@onecoursesource ~]# sysctl -a | head
kernel.sched_child_runs_first = 0
kernel.sched_min_granularity_ns = 1000000
kernel.sched_latency_ns = 5000000
kernel.sched_wakeup_granularity_ns = 1000000
kernel.sched_tunable_scaling = 1
kernel.sched_features = 3183
kernel.sched_migration_cost = 500000
kernel.sched_nr_migrate = 32
kernel.sched_time_avg = 1000
kernel.sched_shares_window = 10000000
```

ExamAlert

You are not expected to know details of any specific configuration for the Linux+ XK0-005 exam.

The name of the parameter (**kernel.sched_child_runs_first**, for example) is a relative pathname that starts from **/proc/sys** and has a dot (**.**) character between the directory and filename rather than a slash (**/**) character. For example, the **/proc/sys/dev/cdrom/lock** file is named using the **dev.cdrom.lock** parameter:

```
[root@onecoursesource ~]# sysctl -a | grep dev.cdrom.lock
dev.cdrom.lock = 1
```

You can change the value of this parameter by using the **sysctl** command:

```
[root@onecoursesource ~]# sysctl dev.cdrom.lock=0
dev.cdrom.lock = 0
[root@onecoursesource ~]# sysctl -a | grep dev.cdrom.lock
dev.cdrom.lock = 0
```

It is actually safer to use the **sysctl** command than to modify the file directly because the **sysctl** command knows which values for the parameter are valid and which ones are not:

```
[root@onecoursesource ~]# sysctl dev.cdrom.lock="abc"
error: "Invalid argument" setting key "dev.cdrom.lock"
```

The **sysctl** command knows which parameter values are valid because it can look at the **modinfo** output. For example, the value of the **lock** file must be a Boolean (0 or 1), according to the output of the **modinfo** command:

```
[root@onecoursesource ~]# modinfo cdrom | grep lock
parm:           lockdoor:bool
```

If you modify the file directly or use the **sysctl** command, the changes are temporary. When the system is rebooted, the values go back to the defaults, unless you make changes in the **/etc/sysctl.conf** file. The next section provides more details about **/etc/sysctl.conf**.

/etc/sysctl.conf

The **/etc/sysctl.conf** file is used to specify which kernel parameters to enable at boot.

Example:

```
[root@OCS ~]# cat /etc/sysctl.conf
# System default settings live in/usr/lib/sysctl.d/00-system.conf.
# To override those settings, enter new settings here, or
# in an /etc/sysctl.d/<name>.conf file.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.ipv4.ip_forward=1
```

In this example, the kernel parameter **ip_forward** is turned on, which means this machine will act as a router between two networks.

There are thousands of possible kernel settings, including dozens of settings that affect networking. The **ip_forward** setting is one of the most common network settings.

The parameters that optimize the IO (input/output) scheduler are examples of kernel parameters. Several parameters can be set to change the behavior of the scheduler. This section covers the parameters that are important for the Linux+ XK0-005exam.

To see the current scheduler, view the contents of the **/sys/block/**<*device*>**/queue/scheduler** file (where <*device*> is the actual device name). Here's an example:

```
[root@OCS ~]# cat /sys/block/sda/queue/scheduler
[noop] deadline cfq
```

The value within the square brackets is the default. To change this, use the **echo** command, as shown here:

```
[root@OCS ~]# echo "cfq" > /sys/block/sda/queue/scheduler
[root@OCS ~]# cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
```

Additional scheduler types include the following:

- **cfq:** The Completely Fair Queuing schedule has a separate queue for each process, and the queues are served in a continuous loop.
- **noop:** This schedule follows the FIFO (first in, first out) principle.
- **deadline:** This is the standard scheduler. This scheduler creates two queues: a read queue and a write queue. It also puts a timestamp on each I/O request to ensure that requests are handled in a timely manner.

Modules

A *module* is a small software program that, when loaded, provides more features and capabilities to the kernel. This section describes the management of modules using the **lsmod**, **imsmod**, **rmmod**, **insmod**, **modprobe**, and **modinfo** commands.

lsmod

The **lsmod** command displays the kernel modules that are loaded into memory. This command has no options.

In the output of the **lsmod** command, each line describes one module. There are three columns of information for each line:

- The module name.
- The size of the module, in bytes.
- The "things" that are using the module. A "thing" could be a filesystem, a process, or another module. In the event that another module is using this module, the dependent module name is listed. Otherwise, a numeric value that indicates how many "things" use this module is provided.

Example:

```
[root@OCS ~]# lsmod | head
Module                  Size       Used by
tcp_lp                  12663      0
bnep                    19704      2
```

```
bluetooth                372944     5   bnep
rfkill                   26536      3   bluetooth
fuse                     87741      3
xt_CHECKSUM              12549      1
ipt_MASQUERADE           12678      3
nf_nat_masquerade_ipv4   13412       1   ipt_MASQUERADE
tun                       27141 1
```

imsmod

The **imsmod** command is used to add modules to the currently running kernel.

Syntax:

```
insmod [module_name]
```

The exact location of the module needs to be specified. For example:

```
[root@OCS ~]# lsmod | grep fat
[root@OCS ~]# insmod /usr/lib/modules/3.19.8-100.fc20.x86_64/kernel/
fs/ fat.ko
[root@OCS ~]# lsmod | grep fat
fat                    65107 0
```

There are no options to the **insmod** command; however, each module might have modules that can be passed into the module using the following syntax:

```
insmod module options
```

The **insmod** command has two disadvantages:

- You have to know the exact location of the module.
- If the module has any dependencies (that is, if the module needs another module), it will fail to load.

rmmod

The **rmmod** command is used to remove modules from the currently running kernel.

Syntax:

```
rmmod [options] [module_name]
```

Example:

```
[root@OCS ~]# lsmod | grep fat
fat   65107 0
[root@OCS ~]# rmmmod fat
[root@OCS ~]# lsmod | grep fat
```

Modules that are currently in use will not be removed by this command by default.

Key options for the **rm** command include the following:

- **-f** attempts to force removal of modules that are in use (which is very dangerous).
- **-w** waits for a module to be no longer used and then removes it.
- **-v** displays verbose messages.

insmod

The **insmod** command is used to add modules to the currently running kernel.

Syntax:

```
insmod [module_name]
```

The exact location of the module needs to be specified. For example:

```
[root@OCS ~]# lsmod | grep fat
[root@OCS ~]# insmod /usr/lib/modules/3.19.8-100.fc20.x86_64/kernel/
fs/ fat.ko
[root@OCS ~]# lsmod | grep fat
fat                    65107 0
```

There are no options to the **insmod** command; however, each module might have modules that can be passed into the module using the following syntax:

```
insmod module options
```

The **insmod** command has two disadvantages:

- You have to know the exact location of the module.
- If the module has any dependencies (that is, if the module needs another module), it will fail to load.

modprobe

The **modprobe** command is used to add and remove modules from the currently running kernel. It also attempts to load module dependencies.

Syntax:

```
modprobe [options] [module_name]
```

When used to remove modules (with the **-r** option), the **modprobe** command also removes dependency modules unless they are in use by another part of the subsystem (such as the kernel or a process).

Key options for the **modprobe** command include the following:

- **-c** displays the current **modprobe** configuration.
- **-q** causes **modprobe** to run in quiet mode.
- **-R** displays all modules that match an alias to assist you in debugging issues.
- **-r** removes the specified module from memory.
- **-v** displays verbose messages; this is useful for determining how **modprobe** is performing a task.

modinfo

The **modinfo** command is used to provide details about a module.

Syntax:

```
modinfo [module_name]
```

Example:

```
[root@OCS ~]# modinfo xen_wdt
filename: /lib/modules/3.19.8-100.fc20.x86_64/kernel/drivers/watchdog/
xen_wdt.ko
license: GPL
version: 0.01
description:   Xen WatchDog Timer Driver
author: Jan Beulich <jbeulich@novell.com>
srcversion:   D13298694740A00FF311BD0
depends:
intree:     Y
```

```
vermagic:       3.19.8-100.fc20.x86_64 SMP mod_unload
signer:          Fedora kernel signing key
sig_key:        06:AF:36:EB:7B:28:A5:AD:E9:0B:02:1E:17:E6:AA:B2:B6:52:
63:AA
sig_hashalgo:   sha256
parm:           timeout:Watchdog timeout in seconds (default=60)(uint)
parm:            nowayout:Watchdog cannot be stopped once started
(default=0) (bool)
```

One of the most important parts of the output of the **modinfo** command is the **parm** values, which describe parameters that can be passed to this module to affect its behavior.

Configure Common System Services

This section focuses on configuring common system services, including SSH, NTP, syslog, chrony, and localization.

SSH

The Secure Shell (SSH) protocol is designed to replace insecure remote communication operations, such as the **telnet**, **ftp**, **rlogin**, **rsh**, **rcp**, and **rexec** commands/protocols. The primary issue with earlier communication methods is that those methods send data across the network in plaintext rather than in an encrypted format. In some cases, such as with **telnet** and **ftp**, this can include sending user account data (such as name and password) across the network in plaintext.

SSH provides a better level of security by encrypting the data sent across the network. SSH has become such a standard in Linux that almost all distributions include both the client and server software by default. In the event that you do not have this software installed on your system, you should install the **openssh**, **openssh-server**, **openssh-clients**, and **openssh-askpass** software packages.

The **/etc/ssh** directory is the location where the Secure Shell configuration files are stored. The configuration file for the SSH server is the **/etc/ssh/sshd_config** file. Don't confuse this with the **/etc/ssh/ssh_config** file, which is used to configure client utilities, such as the **ssh**, **scp**, and **sftp** commands.

There are two different SSH protocols that are numbered 1 and 2. These are not versions but rather two separate protocols developed to provide secure data connections. There was a time when both protocols were commonly used, but

now almost all SSH clients use only protocol 2. To set the protocol that your SSH server accepts, use the **Protocol** keyword:

```
Protocol 2
```

If you have some older SSH clients that require protocol 1, you can configure your SSH server to accept both protocol connections by using the following keyword setting in the **/etc/ssh/sshd_config** file:

```
Protocol 1,2
```

If you have multiple network cards (or virtual interfaces), you may want to limit the SSH server to listen to only some of the network cards. To do this, use the **ListenAddress** keyword and specify the IP address assigned to the network cards that SSH should accept connections on:

```
ListenAddress 192.168.1.100:192.168.1.101
```

The standard port number that the SSH server listens to is port 22. You can modify the SSH server to listen to another port by using the **Port** keyword:

```
Port 2096
```

You might need to change what sort of log messages you want the SSH server to record. This can be set by using the **LogLevel** keyword. The levels available are as follows:

- **QUIET**
- **FATAL**
- **ERROR**
- **INFO**
- **VERBOSE**
- **DEBUG**
- **DEBUG1** (which is the same as **DEBUG**)
- **DEBUG2**
- **DEBUG3**

Network Time Protocol (NTP)

The Network Time Protocol daemon (**ntpd**) is a process that ensures the system clock is in sync with the time provided by remote NTP servers. Most of

the configuration for this process is handled via the **/etc/ntp.conf** file. Table 7.1 shows the important settings of the **/etc/ntp.conf** file.

TABLE 7.1 **/etc/ntp.conf File Settings**

Option	Description
driftfile	Contains a value that represents the typical delta (change) over time from the NTP-reported time and the system clock. This value is used to regularly update the system clock without having to access an NTP server.
restrict	Used to indicate restrictions for the daemon, including what machines can access this NTP server when it is used as a service.
server	Used to list an NTP server for this machine when it is used as an NTP client.

Here is an example of a typical **/etc/ntp.conf** file:

```
# For more information about this file, see the man pages
# ntp.conf(5), ntp_acc(5), ntp_auth(5), ntp_clock(5), ntp_misc(5),
ntp_mon(5).

driftfile /var/lib/ntp/drift

# Permit time synchronization with our time source, but do not
# permit the source to query or modify the service on this system.
restrict default kod nomodify notrap nopeer noquery

# Permit all access over the loopback interface. This could
# be tightened as well, but to do so would effect some of
# the administrative functions.
restrict 127.0.0.1
restrict ::1

# Hosts on local network are less restricted.
# restrict 192.168.1.0 mask 255.255.255.0 nomodify notrap

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.
  html).
server 0.fedora.pool.ntp.org iburst
server 1.fedora.pool.ntp.org iburst
```

```
server 2.fedora.pool.ntp.org iburst
server 3.fedora.pool.ntp.org iburst

# Enable public key cryptography.
#crypto

includefile /etc/ntp/crypto/pw

# Key file containing the keys and key identifiers used when operating
# with symmetric key cryptography.
keys /etc/ntp/keys
```

The **pool.ntp.org** address is a link to a cluster of NTP servers that are geographically spread throughout the world. These servers can be freely used within the **/etc/ntp.conf** file. For example, the following servers are provided by the Fedora project (but note that these are often mirrors, pointing to other systems, so the resulting hostnames for these servers will be different once you have connected to them):

0.fedora.pool.ntp.org

1.fedora.pool.ntp.org

2.fedora.pool.ntp.org

3.fedora.pool.ntp.org

The **ntpq** command allows you to perform queries on NTP servers. For example, the **ntpq** command in the following example displays a summary of the status of NTP servers:

```
[root@onecoursesource ~]# ntpq -p
     remote           refid      st t when poll reach   delay   offset
jitter
==============================================================
*propjet.latt.ne 68.110.9.223    2 u  120 1024  377   98.580    7.067
4.413
-services.quadra 208.75.88.4      3 u  272 1024  377   72.504  -10.689
1.612
+mirror          216.93.242.12    3 u  287 1024  377   20.406   -2.555
0.822
+108.61.194.85.v 200.23.51.102    2 u  741 1024  377   69.403   -3.670
1.610
```

Table 7.2 lists some important options to the **ntpq** command.

TABLE 7.2 **ntpq Command Options**

Option	Description
-d	Enables debugging mode.
-n	Lists host IP addresses rather than names.
-p	Prints a list of all peers.

Syslog

The syslog service has existed since 1980. Although it was advanced at the time it was created, its limitations have grown over time as more complex logging techniques have become required.

In the mid-2000s, the rsyslog service was created as an extension of the traditional syslog service. The rsyslog service extends the capabilities of syslog through the inclusion of modules.

The configuration of syslog and rsyslog services is consistent, with the exception of slightly different naming conventions (for example, **rsyslog.conf** versus **syslog.conf**) and additional features available in the log files.

The **syslogd** or **rsyslogd** daemon is responsible for logging of application and system events. It determines which events to log and where to place log entries, based on configuration settings in the **/etc/syslog.conf** file.

Table 7.3 describes some important options to the **syslogd** and **rsyslogd** commands.

TABLE 7.3 **syslogd and rsyslogd Command Options**

Option	Description
-d	Enables debugging mode.
-f	Specifies the configuration file (with **/etc/syslog.conf** as the default).
-m *x*	Creates a timestamp in the log files every *x* minutes. (You can set *x* to **0** to omit timestamps.)
-r	Enables the **syslogd** daemon to accept logs from remote systems.
-S	Enables verbose mode.
-x	Disables DNS lookups for IP addresses.

The **/etc/rsyslog.conf** file is one of the configuration files for the **rsyslogd** daemon. The following is a typical **rsyslog.conf** file with the comments and blank lines removed (along with the modules):

```
[root@OCS ~]# grep -v "^$" /etc/rsyslog.conf | grep -v "^#"
*.info;mail.none;authpriv.none;cron.none /var/log/messages
authpriv.*                               /var/log/secure
mail.*                                   -/var/log/maillog
cron.*                                    /var/log/cron
*.emerg                                    *
uucp,news.crit                            /var/log/spooler
local7.*                                 /var/log/boot.log
```

ExamAlert

On the Linux+ XK0-005 exam, you might be given an entry line from the **/etc/rsyslog.conf** file and be tested on your understanding of what is logged and where.

Every line represents one logging rule that is broken into two primary parts: the selector (for example, **uucp,news.crit**) and the action (**/var/log/spooler**). The selector is also broken into two parts: the facility (**uucp,news**) and the priority (**crit**). Note that when a priority is provided, it means "this priority and all priorities of a higher level."

The following list shows the available facilities in order from lower level to higher level:

- **auth** (or security)
- **authpriv**
- **cron**
- **daemon**
- **kern**
- **lpr**
- **mail**
- **mark**
- **news**
- **syslog**

- **user**
- **uucp**
- **local0** through **local7**

The following list shows the available priority levels:

- **debug**
- **info**
- **notice**
- **warning** (or **warn**)
- **err** (or **error**)
- **crit**
- **alert**
- **emerg** (or **panic**)

The following list shows the available "actions", which are really just where the log entry should be sent:

- Regular file (where using **-** before the filename prevents syncing with every log entry, thus reducing hard drive writes)
- Named pipes
- Console or terminal devices
- Remote hosts
- Users, which write to the specified user's terminal windows (where ***** specifies all users)

chrony

Traditionally, the NTP server on Linux has been the **ntpd** server (discussed earlier in this chapter, in the "Network Time Protocol [NTP]" section). A newer alternative, chrony, provides some benefits over the **ntpd** server, including:

- Faster synchronization
- Typically more accurate time

- Improved response to clock frequency changes
- No need for periodic polling of other NTP servers
- Smaller (less memory and CPU utilization)

Localization

Several commands can be used to display or modify the date and time on a system, as well as the locale of a system. This section explores these commands.

timedatectl

Use the **timedatectl** command to display the system clock.

Syntax:

```
timedatectl [option] [value]
```

Example:

```
[root@OCS ~]# timedatectl
       Local time   :       Wed 2018-10-10 14:41:41 PDT
       Universal time:      Wed 2018-10-10 21:41:41 UTC
       RTC time:            Wed 2018-10-10 09:51:09
      Timezone:             America/Los_Angeles (PDT, -0700)
      NTP enabled:          yes
       NTP synchronized:     yes
       RTC in local TZ:      no
       DST active:           yes
        Last DST change:    DST began at
                            Sun 2018-03-11 01:59:59 PST
                            Sun 2018-03-11 03:00:00 PDT
      Next DST change:  DST ends (the clock jumps one hour backwards)
at
                            Sun 2018-11-04 01:59:59 PDT
                             Sun 2018-11-04 01:00:00 PST
```

As the root user, you can use this command to set the system clock. Table 7.4 demonstrates the most commonly used methods of changing the system clock.

TABLE 7.4 **Methods to Change the System Clock**

Method	Description
set-time [*time*]	Sets the system clock to the specified *time*.
set-timezone [*zone*]	Sets the system time zone to the specified *zone*.
set-ntp [0|1]	Enables (**1**) or disables (**0**) Network Time Protocol.

localectl

The BASH shell and other processes need customized operations to fit the location of the user. For example, if currency is to be displayed and the user is located in the United States, the $ character should be used. If the user is located in Great Britain, the £ character should be used.

This section focuses on the variables used to inform programs what settings to use based on a user's locale.

LC_* refers to a collection of locale settings used to change the way the shell and other programs handle differences based on the geographic region of the user (or a region the user is familiar with). These values can be viewed by executing the **locale** command:

```
[root@OCS ~]# locale
LANG=en_US.UTF-8
LANGUAGE=en_US
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
```

The most important locale settings are described in Table 7.5.

TABLE 7.5 **Locale Settings**

Setting	Description
LANG	Language
LC_CTYPE	Case conversion
LC_NUMERIC	Numeric format
LC_TIME	Time and date format
LC_COLLATE	Collation order
LC_MONETARY	Currency format
LC_MESSAGES	Format of message
LC_PAPER	Paper size format
LC_NAME	Name format
LC_ADDRESS	Address format
LC_TELEPHONE	Telephone format
LC_ALL	When set, LC_ALL will override all other locale settings. This provides an easy means to change all locale settings by modifying one environment variable.

The **localectl** command can display and change both locale values and keyboard layouts.

Syntax:

```
localectl [options] command
```

To display values, use **status**:

```
[root@OCS ~]# localectl status
 System Locale:  LANG=en_US.utf8
    VC Keymap:   us
    X11 Layout:   us
    X11 Model:    pc105+inet
     X11 Options: terminate:ctrl_alt_bksp
```

Set the locale and keyboard as follows:

```
[root@OCS ~]# localectl set-locale "LANG=de_DE.utf8"set-keymap "de"
```

There are a handful of options to the **localectl** command, but none of them are commonly used.

Cram Quiz

Answer these questions. The answers follow the last question. If you cannot answer these questions correctly, consider reading this chapter again until you can.

1. Which file can be revised to modify kernel parameters during the boot process?
   - ❍ **A. /etc/sysctl.conf**
   - ❍ **B. /etc/kern.parm**
   - ❍ **C. /etc/kern.conf**
   - ❍ **D. /etc/sys.conf**

2. Which command correctly sets the CD-ROM lock kernel parameter to 0?
   - ❍ **A. sysctl dev.cdrom.lock 0**
   - ❍ **B. sysctl dev.cdrom.lock=0**
   - ❍ **C. sysctl dev/cdrom/lock=0**
   - ❍ **D. sysctl dev/cdrom/lock 0**

3. Which of the following commands can be used to remove a module from memory? (Choose two.)
   - ❍ **A. modprobe**
   - ❍ **B. insmod**
   - ❍ **C. modinfo**
   - ❍ **D. rmmod**

4. Based on the following line from the **/etc/rsyslog.conf** file, which levels of messages for the **cron** service are logged?

   ```
   cron.warn                                    /var/log/cron
   ```

   - ❍ **A.** All levels
   - ❍ **B.** Just warn level
   - ❍ **C.** Warn level and levels with higher priorities
   - ❍ **D.** Warn level and levels with lower priorities

Cram Quiz Answers

1. **A.** The **/etc/sysctl.conf** file is used to specify which kernel parameters to enable at boot. The other answers are not valid kernel configuration files.
2. **B.** You can change the value of this parameter by using the **sysctl** command: **sysctl dev.cdrom.lock=0**. The character between **dev**, **cdrom**, and **lock** must be a **.**, not a **/** character. An equal sign must be between the parameter and the value.
3. **A and D.** The **rmmod** command is designed specifically to remove a module from memory. The **-r** option to the **modprobe** command also can be used to remove a module from memory. The **insmod** command inserts a module into memory, and the **modinfo** command provides details about a module.
4. **C.** When a priority is provided, it means “this priority and all priorities of a higher level.”

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

EXAM CRAM

CompTIA® Linux+

XK0-005

CompTIA® Linux+ XK0-005 Exam Cram is an all-inclusive study guide designed to help you pass the updated version of the CompTIA Linux+ exam. Prepare for test day success with complete coverage of exam objectives and topics, plus hundreds of realistic practice questions. Extensive prep tools include quizzes, Exam Alerts, and our essential last-minute review CramSheet. The powerful Pearson Test Prep practice software provides real-time assessment and feedback with two complete exams.

Covers the critical information needed to score higher on your Linux+ XK0-005 exam!

- Manage files and directories
- Configure and manage storage
- Manage software configurations
- Implement identity management
- Implement and configure firewalls
- Create simple shell scripts to automate common tasks
- Perform basic container operations
- Analyze and troubleshoot storage issues and network resource issues

Prepare for your exam with Pearson Test Prep

- Realistic practice questions and answers
- Comprehensive reporting and feedback
- Customized testing in study, practice exam, or flash card modes
- Complete coverage of Linux+ XK0-005 exam objectives

At the impressionable age of 14, WILLIAM "BO" ROTHWELL crossed paths with a TRS-80 Micro Computer System (affectionally known as a "Trash 80"). Soon after, the adults responsible for Bo made the mistake of leaving him alone with the TSR-80. He immediately dismantled it and held his first computer class, showing his friends what made this "computer thing" work. Since that experience, Bo's passion for understanding how computers work and sharing this knowledge with others has resulted in a rewarding career in IT training. His experience includes Cloud, Linux, Unix, IT security, DevOps, and programming languages such as Perl, Python, Tcl, and BASH. He is the founder and lead instructor of One Course Source, an IT training organization.

Shelving Category: Certification
Covers: CompTIA Linux+ XK0-005 exam

www.pearsonITcertification.com

Pearson

COMPANION WEBSITE

Your purchase includes access to the practice exams in multiple test modes and the CramSheet.

Includes Exclusive Offer for up to **80% Off** Premium Edition eBook and Practice Tests

Pearson Test Prep online system requirements:
Browsers: Chrome version 73 and above; Safari version 12 and above; Microsoft Edge 44 and above.
Devices: Desktop and laptop computers, tablets running Android v8.0 and above or iPadOS v13 and above, smartphones running Android v8.0 and above or iOS v13 and above with a minimum screen size of 4.7". Internet access required.

Pearson Test Prep offline system requirements:
Windows 10, Windows 8.1; Microsoft .NET Framework 4.5 Client; Pentium-class 1 GHz processor (or equivalent); 512 MB RAM; 650 MB disk space plus 50 MB for each downloaded practice exam;

ISBN-13: 978-0-13-789855-8
ISBN-10: 0-13-789855-X

U.S. $49.99