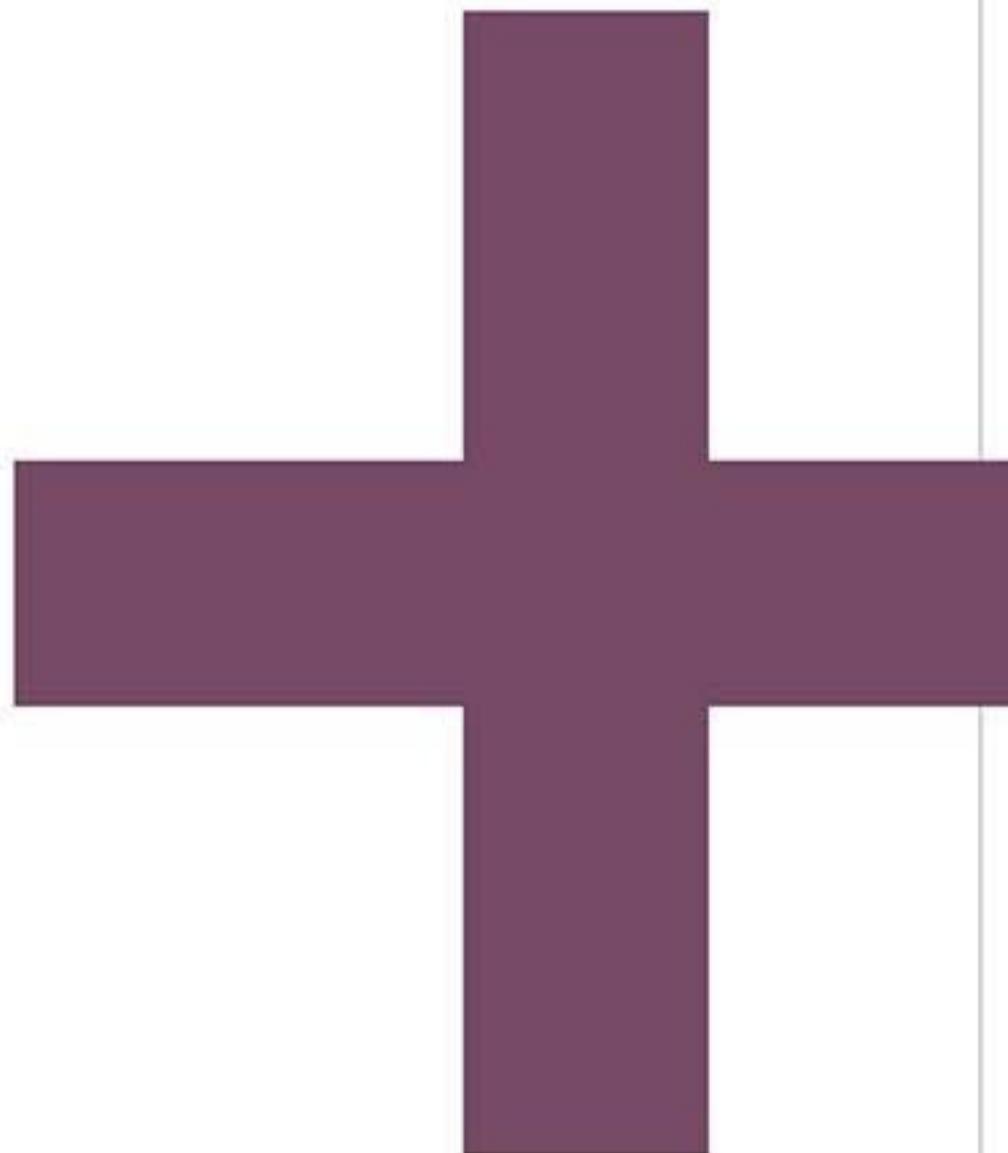


CompTIA



The Official CompTIA
Linux+
Student Guide

Exam XKO-004



Official CompTIA Content Series for CompTIA Performance Certifications

Table of Contents

Lesson 1: Performing Basic Linux Tasks	1
Topic A: Identify the Linux Design Philosophy	2
Topic B: Enter Shell Commands	12
Topic C: Get Help with Linux.....	24
Lesson 2: Managing Users and Groups.....	35
Topic A: Assume Superuser Privileges	36
Topic B: Create, Modify, and Delete Users	44
Topic C: Create, Modify, and Delete Groups.....	56
Topic D: Query Users and Groups.....	62
Topic E: Configure Account Profiles.....	68
Lesson 3: Managing Permissions and Ownership.	75
Topic A: Modify File and Directory Permissions	76
Topic B: Modify File and Directory Ownership.....	87
Topic C: Configure Special Permissions and Attributes	93
Topic D: Troubleshoot Permissions Issues.....	105
Lesson 4: Managing Storage.....	113
Topic A: Create Partitions.....	114
Topic B: Manage Logical Volumes.....	130
Topic C: Mount File Systems	138
Topic D: Manage File Systems	144
Topic E: Navigate the Linux Directory Structure.....	154
Topic F: Troubleshoot Storage Issues.....	162
Lesson 5: Managing Files and Directories.	177
Topic A: Create and Edit Text Files.	178
Topic B: Search for Files.....	190
Topic C: Perform Operations on Files and Directories.....	198

Topic D: Process Text Files.....	209
Topic E: Manipulate File Output.....	226
Lesson 6: Managing Kernel Modules.....	237
Topic A: Explore the Linux Kernel	238
Topic B: Install and Configure Kernel Modules.....	245
Topic C: Monitor Kernel Modules	253
Lesson 7: Managing the Linux Boot Process.....	261
Topic A: Configure Linux Boot Components	262
Topic B: Configure GRUB 2	274
Lesson 8: Managing System Components.....	283
Topic A: Configure Localization Options	284
Topic B: Configure GUIs	295
Topic C: Manage Services.....	308
Topic D: Troubleshoot Process Issues.....	320
Topic E: Troubleshoot CPU and Memory Issues.....	334
Lesson 9: Managing Devices.....	345
Topic A: Identify the Types of Linux Devices.....	346
Topic B: Configure Devices.....	354
Topic C: Monitor Devices.....	364
Topic D: Troubleshoot Hardware Issues.....	370
Lesson 10: Managing Networking	381
Topic A: Identify TCP/IP Fundamentals.....	382
Topic B: Identify Linux Server Roles	390
Topic C: Connect to a Network	398
Topic D: Configure DHCP and DNS Client Services	412
Topic E: Configure Cloud and Virtualization Technologies	420
Topic F: Troubleshoot Networking Issues	432

Lesson 11: Managing Packages and Software	453
Topic A: Identify Package Managers.	454
Topic B: Manage RPM Packages with YUM	459
Topic C: Manage Debian Packages with APT.....	465
Topic D: Configure Repositories	470
Topic E: Acquire Software.....	476
Topic F: Build Software from Source Code.....	481
Topic G: Troubleshoot Software Dependency Issues	486
Lesson 12: Securing Linux Systems.	491
Topic A: Implement Cybersecurity Best Practices.	492
Topic B: Implement Identity and Access Management Methods	507
Topic C: Configure SELinux or AppArmor	520
Topic D: Configure Firewalls.....	529
Topic E: Implement Logging Services.....	543
Topic F: Back Up, Restore, and Verify Data	553
Lesson 13: Working with Bash Scripts.	577
Topic A: Customize the Bash Shell Environment	578
Topic B: Identify Scripting and Programming Fundamentals.....	588
Topic C: Write and Execute a Simple Bash Script	595
Topic D: Incorporate Control Statements in Bash Scripts	605
Lesson 14: Automating Tasks.....	619
Topic A: Schedule Jobs.....	620
Topic B: Implement Version Control Using Git.....	627
Topic C: Identify Orchestration Concepts.....	634
Lesson 15: Installing Linux.....	639
Topic A: Prepare for Linux Installation.....	640
Topic B: Perform the Installation	647

Appendix A: Mapping Course Content to CompTIA® Linux+® (Exam XK0-004)	657
Solutions	677
Glossary	739
Index	753

About This Course

For many years, Linux has dominated the server install base in the business world—and it will continue to do so in the foreseeable future. Linux's popularity has led to a greater need for information technology (IT) professionals who can manage servers that run some form of the Linux kernel. *The Official CompTIA® Linux+®* courseware builds on your existing experience with systems operations and administration to provide you with the knowledge and skills required to configure, manage, operate, and troubleshoot a Linux environment by using security best practices, scripting, and automation.

Course Description

Target Student

This course is designed for IT professionals whose primary job responsibility is the management of servers and other devices running the Linux operating system. A typical student in this course should have at least nine months of hands-on Linux experience and at least one and a half years of IT experience in other computing environments. The target student should wish to expand their skillset to support their career in Linux system administration and operation.

This course is also designed for students who are seeking the CompTIA Linux+ certification and who want to prepare for Exam XK0-004. The Linux+ certification can validate the student's understanding and skill in configuring, monitoring, and supporting Linux systems.

Course Prerequisites

To ensure your success in this course, you should have at least foundational experience with general systems administration procedures, some hands-on exposure to one or more Linux distributions, as well as knowledge of computing hardware and basic networking and cybersecurity concepts.

You can obtain this level of skills and knowledge by taking the following official CompTIA courses:

- *The Official CompTIA® A+® Core 1 and 2 Student Guide (Exams 220-1001 and 220-1002)*
- *The Official CompTIA® Network+® Student Guide (Exam N10-007)*
- *The Official CompTIA® Security+® Student Guide (Exam SY0-501)*

Note: These prerequisites might differ significantly from the prerequisites for the CompTIA certification exams. For the most up-to-date information about the exam prerequisites, complete the form on this page: <https://certification.comptia.org/training/exam-objectives>

Course Objectives

In this course, you will configure, operate, and troubleshoot Linux systems. You will:

- Perform basic Linux tasks.
- Manage users and groups.
- Manage permissions and ownership.
- Manage storage.
- Manage files and directories.
- Manage kernel modules.
- Manage the Linux boot process.
- Manage system components.
- Manage devices.
- Manage networking.

- Manage packages and software.
- Secure Linux systems.
- Write and execute Bash shell scripts.
- Automate tasks.
- Plan and perform a Linux installation.

How to Use This Book

As You Learn

This book is divided into lessons and topics, covering a subject or a set of related subjects. In most cases, lessons are arranged in order of increasing proficiency.

The results-oriented topics include relevant and supporting information you need to master the content. Each topic has various types of activities designed to enable you to solidify your understanding of the informational material presented in the course.

Information is provided for reference and reflection to facilitate understanding and practice.

Data files for various activities as well as other supporting files for the course are available by download from the course website. In addition to sample data for the course exercises, the course files may contain media components to enhance your learning and additional reference materials for use both during and after the course.

At the back of the book, you will find a glossary of the definitions of the terms and concepts used throughout the course. You will also find an index to assist in locating information within the instructional components of the book. In many electronic versions of the book, you can click links on key words in the content to move to the associated glossary definition, and on page references in the index to move to that term in the content. To return to the previous location in the document after clicking a link, use the appropriate functionality in your PDF viewing software.

As You Review

Any method of instruction is only as effective as the time and effort you, the student, are willing to invest in it. In addition, some of the information that you learn in class may not be important to you immediately, but it may become important later. For this reason, we encourage you to spend some time reviewing the content of the course after your time in the classroom.

As a Reference

The organization and layout of this book make it an easy-to-use resource for future reference. Taking advantage of the glossary, index, and table of contents, you can use this book as a first source of definitions, background information, and summaries.

Course Icons

Watch throughout the material for the following visual cues.

Student	Student Icon Descriptive
	A Note provides additional information, guidance, or hints about a topic or task.
	A Caution note makes you aware of places where you need to be particularly careful with your actions, settings, or decisions, so that you can be sure to get the desired results of an activity or task.
	Video notes show you where an associated video is particularly relevant to the content. These videos can be accessed through the course website.
	Additional Practice Questions are available in the Assessment section on course website.

Lesson 1

Performing Basic Linux Tasks

LESSON TIME: 1 HOUR, 30 MINUTES

LESSON INTRODUCTION

There is a great amount of depth to the Linux® operating system. Rather than dive right into the specifics, you'll do well to get a high-level overview of what it is you're about to work with. Also, by operating Linux in some fundamental ways, you'll be better prepared for the journey ahead.

LESSON OBJECTIVES

In this lesson, you will:

- Identify the high-level design concepts that make up the Linux operating system.
- Use fundamental Linux shell commands to get started with the command-line interface (CLI).
- Use various resources to find help on the Linux operating system.

Topic A

Identify the Linux Design Philosophy

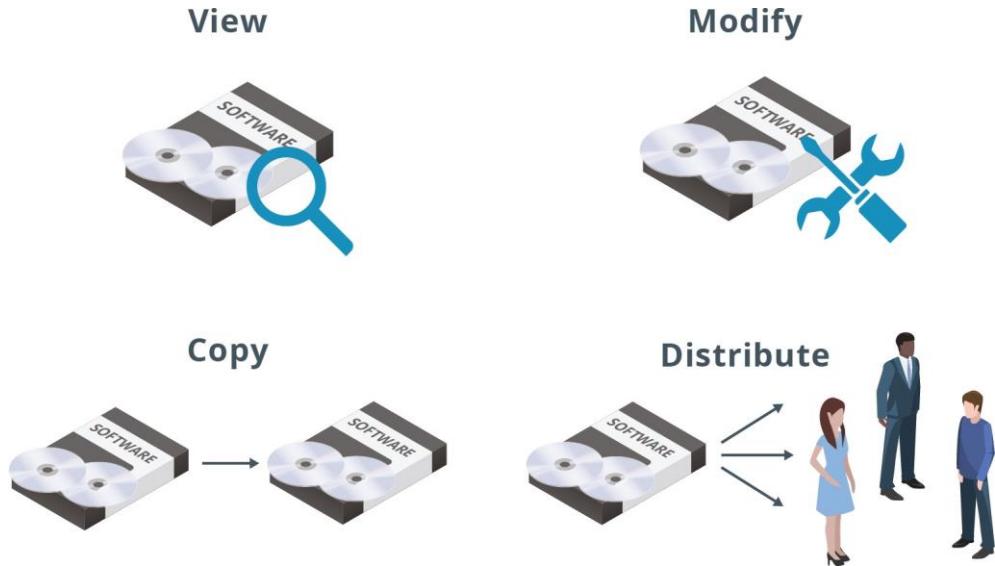
Unlike other operating systems you may have worked with, the history of Linux is driven by a philosophy of openness and transparency. This philosophy has informed the design of Linux as an operating system, as well as its application in the business world. It's important for you to understand this philosophy and how it will ultimately have an impact on your day-to-day operations.

OPEN SOURCE SOFTWARE

Open source software (OSS) refers to computer code that any user is permitted to view, copy, and modify for any reason, as well as distribute to anyone. Users are granted these rights when the author of the software releases the source code under one of several open source licenses. The opposite of OSS is proprietary software—software that is released under a license that imposes restrictions on one or more of the rights just mentioned (view, copy, modify, distribute).

OSS provides several advantages to users, administrators, and programmers alike. Perhaps the most important element of OSS is that it encourages the ongoing improvement of software in a collaborative, community-driven environment.

Individuals or groups of developers may build upon another developer's work to create enhanced or customized software, all while avoiding legal issues. There are many examples of OSS, one of which is the Linux kernel.

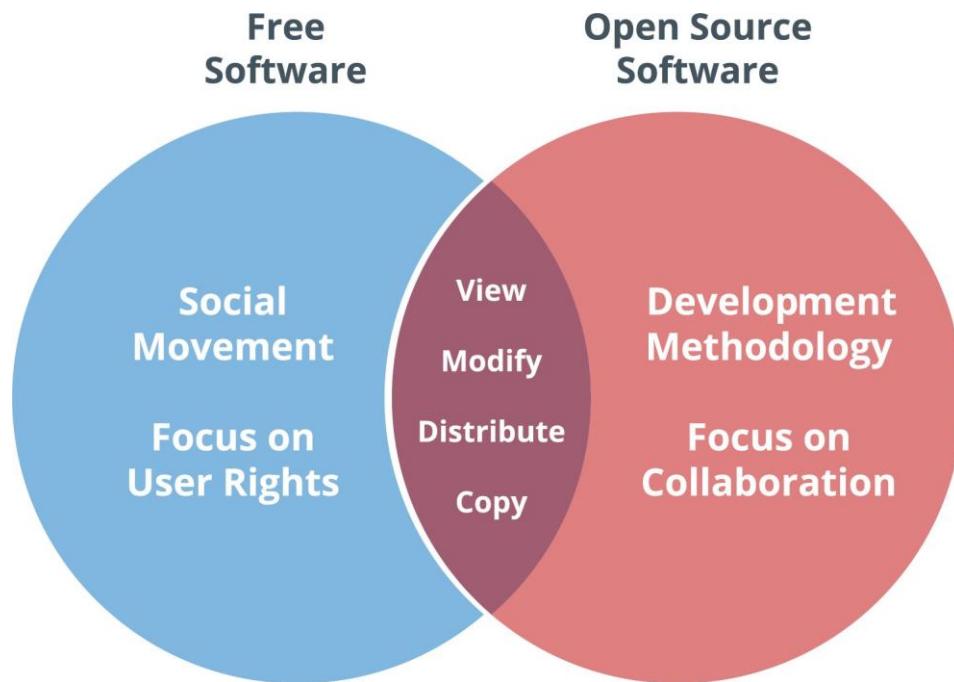


The fundamental permissions granted by open source software.

FREE SOFTWARE VS. OPEN SOURCE SOFTWARE

The term **free software** is often used interchangeably with OSS, and both share a great deal of overlap. However, some draw a distinction between the two. Richard Stallman, the founder of the free software movement, argues that the terms share different values—in his words, "Open source is a development methodology; free software is a social movement." In other words, OSS is most concerned with fostering open collaboration, whereas free software is most concerned with upholding users' rights. Note that "free" is being used in the sense of freedom, not in the sense of having no monetary cost.

In an attempt to resolve this terminology dispute, some developers have adopted the term **free and open source software (FOSS)** to describe any project that embodies the values of both movements.



A Venn diagram of free software and open source software.

FREE AND OPEN SOURCE LICENSES

There are several FOSS licenses that a developer can release their software under. These licenses may differ when it comes to additional restrictions or compatibility with other licenses. The following are some examples of FOSS licenses:

- Apache License
- Berkeley Software Distribution (BSD) license family
- Creative Commons Zero (CC0)
- Eclipse Public License (EPL)
- GNU General Public License (GPL)
- Massachusetts Institute of Technology (MIT) License
- Mozilla Public License (MPL)

THE GNU PROJECT

The **GNU Project** is a free software project led by Richard Stallman. Announced in 1983, it was the first project of its kind, and effectively launched the free software movement. The GNU Project's goal is to create an operating system that is composed

of entirely free software. By the early 1990s, the GNU Project had not completed a full OS, but had created several programs. Stallman wrote much of the GNU software himself, including the GNU C Compiler (GCC) and the Emacs text editor. Later, several programmers worked together to develop more utilities that are compatible with GNU software.

Stallman chose the recursive acronym "GNU's Not Unix" to show that GNU software was similar in design to the proprietary Unix operating system, but did not actually contain any Unix code.



Note: The "G" in GNU is included in the pronunciation of the term "guh-NOO."

FREE SOFTWARE FOUNDATION AND THE GNU GPL

Shortly after kicking off the GNU Project, Stallman founded the nonprofit **Free Software Foundation (FSF)** to promote the philosophy behind free software. Stallman and the FSF created the **GNU General Public License (GPL)** to support the GNU Project. One of the requirements of the GPL is that any derivative of a GPL work must also be distributed under that same license—a concept known as **copyleft**.

THE UNIX PHILOSOPHY

Because GNU software is based on the design of Unix®, it tends to conform to the Unix philosophy. The **Unix philosophy** is a set of best practices and approaches to software development that emphasize simplicity and modularity. This philosophy was created by the lead developers of the Unix operating system and has been summarized in many ways. Some of the key high-level points include:

- **Do one thing and do it well.** Rather than writing a monolithic software tool that accomplishes a variety of disparate tasks, write individual tools that fulfill a specific function.
- **Worse is better.** Software that is limited in functionality ("worse") is often preferable ("better") because it tends to be easier to use and maintain.
- **Support interactivity.** Write individual tools so that they work well with other tools.
- **Handle input/output streams.** Feeding one tool's output as input into another tool is a universally desirable feature.

THE LINUX OPERATING SYSTEM FAMILY

Linux is a family of operating systems based on the Linux kernel, the central core of the OS that manages all components in the system. The Linux kernel was developed by Finnish programmer Linus Torvalds in 1991, while he was a student at the University of Helsinki.

FOSS, the GNU Project, and the Unix design philosophy have all played an important role in the development of Linux. Torvalds released the Linux kernel under version 2 of the GPL. Most distributions of Linux add GNU software written by Stallman and other free software components on top of the Linux kernel. In other words, Linux is the first complete operating system family to qualify as FOSS, and like GNU software, it follows the principles of simplicity and modularity set forth in the Unix design philosophy.



Fundamentally, Linux is a combination of the Linux kernel and GNU software.

GNU/LINUX

Because most members of the Linux OS family incorporate GNU utilities along with the Linux kernel, the FSF prefers to call the OS family "GNU/Linux" rather than just "Linux." This supports the idea that the Linux kernel was the "missing piece" to the GNU Project, and gives credit to GNU for its tools and the free software movement itself.

However, Torvalds and others disagree with this assessment, and the name "Linux" is by far the most common way of referring to the OS family.

ADVANTAGES OF USING LINUX

To summarize, the following are some of the major advantages of using Linux:

- Its FOSS nature promotes transparency.
- Its design emphasizes simplicity and modularity.
- It is highly customizable.
- It is highly reliable and stable.
- It has strong integration with several major programming languages, like C, C++, Python®, Ruby, etc.
- It places an emphasis on security and privacy.
- Most distributions are free of monetary cost.
- It has a passionate community willing to provide support.

DISADVANTAGES OF USING LINUX

No system is perfect, including Linux. The following are some potential disadvantages:

- It has a sharper learning curve than other general purpose operating systems like Windows® and macOS®.
- Desktop software is not as well-supported as it is in other operating systems like Windows and macOS.
- There is no definitive or official version, which can be confusing to new users.
- With some exceptions, there is no official vendor-provided support.

LINUX DISTRIBUTIONS

As a family of operating systems, there is no official OS called "Linux." Instead, there are distinct members of the family called **Linux distributions**, or **distros**. All Linux distros are based on the Linux kernel; they differ primarily in what additional software they add on top of the kernel to create a fully functional OS, as well as the version of the kernel they run. There are also differences in community, rate of release, and other factors. Choosing a distribution is a matter of identifying which one most closely aligns with your business needs as well as your familiarity with its tools.

LIST OF LINUX DISTRIBUTIONS

There are hundreds of distros available. The following table includes some of the most historic and/or popular ones.

Distro	Notes
Slackware	This is the oldest distro that is still actively maintained, initially released in 1993. It is intended to be the most "Unix-like" distro and is most appropriate for advanced Linux users.
Debian	This is another early distro, and one that is composed entirely of free software. It is also the basis for many other derivative distros.
Ubuntu®	This is one of the most popular distros for general use and is based on Debian.
Kali Linux	This Debian-based distro is used for penetration testing and other cybersecurity tasks.
Red Hat® Enterprise Linux® (RHEL)	Unlike most distros that are available free of charge, RHEL comes with paid customer support provided by Red Hat, Inc. This distro is common for servers in corporate environments.
CentOS®	This distro is essentially the no-cost version of RHEL. The functionality is nearly identical, but no customer support is provided.
Fedora®	This distro is sponsored by Red Hat and is the upstream source of RHEL. It has multiple editions that each target different computing roles, and tends to run the latest version of the Linux kernel.
openSUSE	This distro is sponsored by SUSE Linux GmbH and targets several computing roles. It strives to incorporate only tools that qualify as FOSS.
SUSE® Linux Enterprise Server (SLES)	This distro shares a code base with openSUSE, but is intended to be more stable than openSUSE. Like RHEL, SLES comes with paid customer support.
Arch Linux™	This distro is targeted at experienced users and focuses on simplicity as its driving philosophy. Its online documentation repository, ArchWiki, is one of the most comprehensive sources of Linux documentation.

Note: Many Linux distros include proprietary software and are not entirely FOSS.





Note: For more information about Linux distros, visit <https://distrowatch.com>.

MORE ON CENTOS

The CentOS Linux distribution is a stable, predictable, manageable, and reproducible platform derived from the sources of RHEL. CentOS is maintained by the CentOS Project, a community-driven free software effort that has its own governing board. The members of the CentOS Project work independently of the RHEL team. However, CentOS benefits from Red Hat's ongoing contributions and investment, and the CentOS trademark is owned by Red Hat.

This course uses CentOS because it provides a free enterprise-class computing platform that aims to be functionally compatible with the upstream product (RHEL) that it derives from. CentOS does not contain Red Hat's product or certifications, although it is built from the same sources as the upstream enterprise products. More details about this are available in the CentOS FAQ here: <https://wiki.centos.org/FAQ/ General>.

For production environments, the licensed and fully supported RHEL product is recommended.

USES FOR LINUX

One of the main advantages of Linux is that it is highly extensible. As a result, Linux has been applied to many different computing roles. The following table describes these roles.

Role	Notes
Servers	Servers provide an array of services to multiple users over a network. Market share figures for servers are not always easy to pin down, but most sources agree that Linux dominates the server market. More than 90% of the top one million domains are run on Linux web servers, and more than 90% of public cloud servers on the Amazon Elastic Compute Cloud (EC2) platform run Linux.
Workstations	Workstations are more powerful versions of a home desktop and are typically geared toward technical work such as software development. Linux is more common on workstations than it is on PCs, but it still lags behind Windows and macOS in this area.
Mainframes and supercomputers	Mainframes are large, highly advanced computers that excel at processing many transactions at once. Supercomputers are machines at the cutting edge of computing technology that excel at performing complex calculations as quickly as possible. Linux dominates both of these markets—the top 500 fastest supercomputers in the world as of 2018 all run Linux.
Mobile devices	Mobile devices typically include smartphones and tablets, or any device that is small enough to fit in the hand. The mobile operating system Android™ is based on the Linux kernel and has over 80% of the global market share for mobile devices.

Role	Notes
Personal computers (PC)	This typically includes desktops and laptops, or any non-mobile device that is geared toward non-technical home use. This is one of the few markets where Linux has seen minimal penetration; around 2% of users run Linux as a desktop platform.
Embedded systems	Embedded systems are computer hardware and software systems that have a specific function within a larger system. These larger systems can include everything from home appliances like microwaves to large industrial machines. Embedded systems running Linux comprise the majority of the market.

Activity 1-1

Discussing the Linux Design

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **What is the distinction between free software and open source software given by Richard Stallman, the founder of the free software movement?**
2. **What is one of the main advantages of Linux that has enabled it to be applied to many different computing roles?**
3. **In which four major computing roles does Linux dominate the market?**
4. **True or false? There is a single Linux master distribution from which all other Linux sub-distributions are derived.**
5. **Which Unix philosophy states that developers should create software that is limited in functionality because it's easier to use and to maintain?**

Activity 1-2

Identifying the Linux Design

SCENARIO

You are a systems administrator for Develetech Industries, a manufacturer of home electronics located in the fictitious city and state of Greene City, Richland (RL). Your chief technology officer (CTO) is considering switching the company's servers to a Linux platform. He has gathered the team and wants to start a discussion about the nature of Linux, as well as its potential advantages and disadvantages.

1. Linux incorporates software tools from which of the following sources?

- Unix
- The GNU Project
- Berkeley Software Distribution (BSD)
- macOS

2. Which of the following statements about free and open source software (FOSS) are true? (Choose two.)

- Anyone can access the source code.
- The author cannot charge a price for the source code.
- Anyone can modify the source code.
- The source code can only be distributed to a public developer community.

3. Which of the following FOSS licenses is the Linux kernel released under?

- GNU General Public License (GPL) version 2
- GNU GPL version 3
- MIT License
- Apache License 2.0

4. Which of the following best describes a typical Linux distribution?

- The Linux kernel with proprietary software tools.
- The Linux kernel with GNU software tools.
- The Linux kernel only.
- GNU software tools only.

5. True or false? Linux distributions can include proprietary software by default.

- True
- False

6.

Which of the following Linux distributions is the no-cost version of Red Hat Enterprise Linux (RHEL)?

- Ubuntu
- Fedora
- openSUSE
- CentOS

7.

Which of the following characteristics are emphasized by the Unix design philosophy? (Choose two.)

- Modularity
- Stability
- Complexity
- Simplicity

8. True or false? Linux is Unix-like in its design.

- True
- False

9. Which of the following computing roles does Linux dominate in market share? (Choose three.)

- Workstations
- Web servers
- Mobile devices
- Supercomputers

Topic B

Enter Shell Commands

The design of the Linux operating system emphasizes a particular kind of user interface; one in which the user types text commands into a prompt in order to interact with the system. This differs from the primarily visual operating systems like Windows and macOS. Therefore, one of the most crucial skills in Linux administration is becoming comfortable at entering text commands. In this topic, you'll enter a few basic commands to become more familiar with the process.

THE CLI

The **command-line interface (CLI)** is a text-based interface between the user and the operating system that accepts input in the form of commands. The CLI presents a command prompt to the user, and the user enters a command to interact with the system in a variety of ways. Working at the command-line is an important tool in any administrator's arsenal. Developers and administrators often use the CLI, whereas regular users will typically rely on a graphical user interface (GUI).

Comfort at the command-line is essential for administrators. Command-line administration is an assumed skill in Linux. The GUI is not a required component of Linux. In fact, Linux includes many GUIs. Any or none of those may be installed on the system. Assuming you want to maximize the use of hardware for the system's specified purpose, you can perform all administration at the CLI.



Note: In Linux, the CLI is case-sensitive.

CLI ADVANTAGES AND CHALLENGES

Some advantages to using the CLI include:

- It's faster for the system to process.
- It's faster for administrators to enter information.
- Commands can be stored in text files called scripts that you can execute with one command, resulting in a long series of activities by the system.
- Scripts can be scheduled to execute on a regular basis.
- Additional options are available in the CLI that may not be present in the GUI.

Likewise, there are some challenges to using the CLI:

- It's more difficult to learn than a GUI.
- Commands have many options and are not always consistent.
- It's often mistakenly thought of as legacy.
- There are many command-line environments among the Linux, Unix, macOS, and Windows platforms.

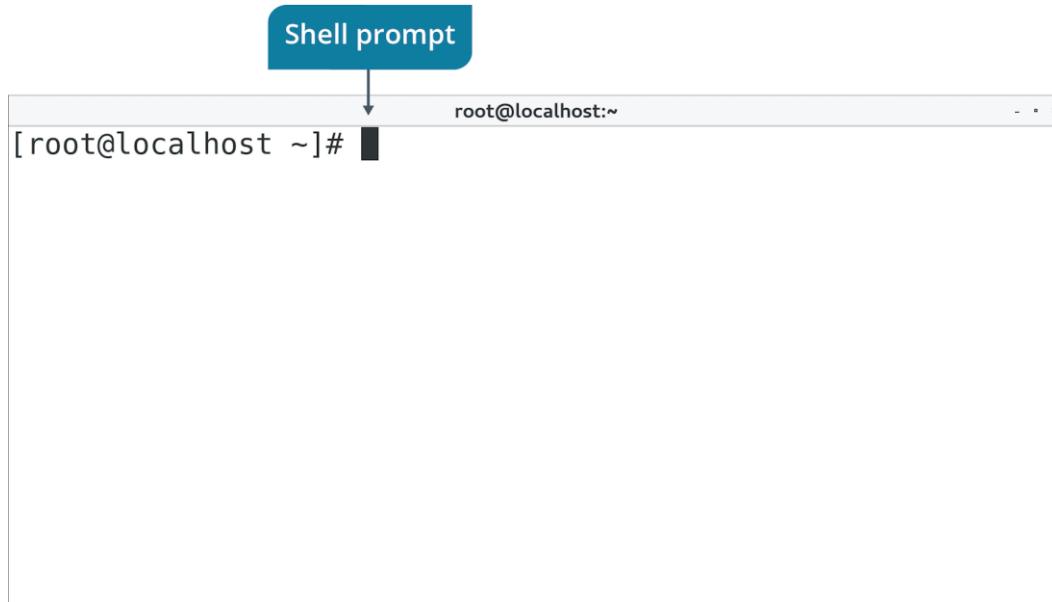
SHELLS

A **shell** envelops the core portion of the operating system—referred to as the kernel—permitting the user to pass commands and information to the kernel. The kernel is also able to respond back to the user via the shell. The shell can be thought of as an

interpreter between human and kernel languages. Linux users issue commands in the shell to tell the operating system what to do. The operating system responds back to the user with the results.

A shell can be implemented as either a CLI or a graphical user interface (GUI). The following table lists some common CLI-based shells that are used in Linux.

Shell	Description
Bourne shell (sh)	This is the original Unix shell that is still available on Linux systems, though not widely used.
Bash (bash)	This is the default Linux shell and a replacement for the Bourne shell. Its full name comes from the term <i>Bourne-again shell</i> .
C shell (csh)	This shell is based on the C programming language and was designed to support C language development environments.
KornShell (ksh)	This shell uses the features of the C shell with the syntax of the Bourne shell. It is common on Unix systems.



A shell prompt in a GUI.

MORE ON BASH

As a Linux user, it is essential to be comfortable using the default Bash shell. Virtually every Linux distribution will use this shell as the translator between the user and the system. It is possible to install and use other shells in Linux if users are more comfortable with them, but the default will almost always be Bash.

Some characteristics of Bash include:

- It's been around for a very long time, so it is well documented with many existing scripts.
- It's commonly used in Linux and macOS (where it is also the default) and with various other operating systems.
- It's not always consistent, since there have been a great many modifications by various groups since its creation.
- It includes history and tab completion features.

- It's very flexible.

BASH SYNTAX

Command-line administration includes the idea of "syntax," or the proper way of structuring a command and any supporting information. The many CLIs have their own unique ways of entering information. You need to understand this syntax to be able to effectively communicate with the interface.

Bash shell syntax contains three main components: the command, options to modify the command, and an argument for the command to act upon. It is common for new users to Bash to forget the spaces between the three components.

The basic syntax of Bash is therefore: **command [-options] [arguments]**



Note: For command syntax, this course uses the convention by which square brackets denote an optional component and curly braces denote a required component. In some cases, a command can be run without options or arguments and still work.

The following table lists an example of each type of basic syntax format using the **ls** command.

Syntax	Command	Description
Command only	ls	Lists directory contents with default output.
Command with options	ls -la	Lists directory contents in long format (-l) and showing "hidden" files (-a).
Command with an argument	ls /var/log	Lists directory contents of /var/log directory with default output.
Command with options and an argument	ls -la /var/log	Lists directory contents of /var/log directory in long format and showing "hidden" files.

```
[root@localhost ~]# ls -la /usr
total 304
drwxr-xr-x. 13 root root 155 Dec 11 13:15 .
dr-xr-xr-x. 18 root root 258 Dec 11 13:51 ..
dr-xr-xr-x.  2 root root 57344 Dec 12 08:41 bin
drwxr-xr-x.  2 root root   6 Apr 11 2018 etc
drwxr-xr-x.  2 root root   6 Apr 11 2018 games
drwxr-xr-x. 10 root root 4096 Dec 11 13:21 include
dr-xr-xr-x. 44 root root 4096 Dec 11 13:21 lib
dr-xr-xr-x. 154 root root 98304 Dec 11 13:37 lib64
drwxr-xr-x. 45 root root 12288 Dec 11 13:21 libexec
drwxr-xr-x. 12 root root 131 Dec 11 13:15 local
dr-xr-xr-x.  2 root root 20480 Dec 12 08:41 sbin
drwxr-xr-x. 257 root root 8192 Dec 11 13:21 share
drwxr-xr-x.  4 root root   34 Dec 11 13:15 src
lrwxrwxrwx.  1 root root    10 Dec 11 13:15 tmp -> ../../var/tmp
[root@localhost ~]#
```

The **ls -la** command displaying the list of files in the **/usr** directory.

ERRORS

If you fail to enter a command in the proper syntax, Bash will return an error. Typically, these error messages are descriptive and will help you to understand what Bash expects. For "command not found" errors, check for typos in the command. For "no such file or directory" errors, check for typos in the directory, file, or file path names.

BASIC BASH COMMANDS

There are several commands that are important for accomplishing basic tasks. Many of these commands will be covered in greater depth throughout the course but are included in this topic to provide basic command vocabulary and to get you started with hands-on practice at the Bash shell.

Command	Description	Examples
echo	Repeats input back to the user on the screen. Commonly used to send information to the user in a script.	echo 'Good morning!' returns "Good morning!" at the CLI.
ls	Lists the contents of a directory. Can be given options to view permissions, hidden files, etc.	<ul style="list-style-type: none"> <code>ls</code> lists contents of current directory. <code>ls -a</code> includes hidden files. <code>ls -l</code> outputs in long format. <code>ls /var/log</code> lists contents of specified directory.
pwd	Displays the current working directory you are in.	<code>pwd</code> returns the path to your current working directory.
cd	Changes your current working directory.	<ul style="list-style-type: none"> <code>cd /var/log</code> changes your current directory to <code>/var/log</code> <code>cd /etc</code> changes your current directory to <code>/etc</code>
touch	Updates timestamp on an existing file, but can also be used to create an empty file.	<code>touch file1</code> updates the timestamp on <code>file1</code> if it exists; creates <code>file1</code> if it doesn't.
cp	Copies a file or directory to another location.	<code>cp file1 file2</code> copies the contents of <code>file1</code> to <code>file2</code> .
mkdir	Creates a directory.	<code>mkdir newdir</code> creates a new directory called <code>newdir</code> .

FILE VIEWING COMMANDS

Linux system configurations are held in text files, so you'll need to be able to view the contents of those files.

The `cat` command is used to view the contents of a file without the option to edit that file. An example of using `cat` is `cat file1` to show the contents of `file1` on the screen.

The `less` command is used to view the contents of a file when those contents won't fit entirely on one screen. This command breaks the content output into pages that you can scroll through at the CLI. An example of using `less` is `less file1` to break

the contents of **file1** into multiple pages when its contents are lengthy enough to go past a single screen. Press **Page Up** and **Page Down** to scroll screens, and press **q** to exit the command.

FILE EDITING COMMANDS

Just as you'll need to view the contents of text files, you'll also need to edit them.

Command	Description	Examples
vim	Starts a powerful text editor and the default for Linux.	<ol style="list-style-type: none"> 1. vim file1 to open a text file in command mode. 2. Press i to enter insert mode. 3. Press Esc to leave insert mode. 4. :Wq to save the file and quit.
nano	Starts a simple, user-friendly text editor. It may not be installed on all distros.	<ol style="list-style-type: none"> 1. nano file1 to open a text file. 2. Enter text directly in the interface. 3. Press Ctrl+O to save changes. 4. Press Ctrl+X to quit.
gedit	Starts a GUI text editor that is easy to use. Requires a desktop environment to be installed.	<ol style="list-style-type: none"> 1. Select Applications→Accessories→Text Editor. 2. Enter text directly in the interface. 3. Use the menu to save and quit.

POWER MANAGEMENT COMMANDS

Periodically, it may be necessary to reboot or shut down the system. There are several commands to accomplish this, but for now you will focus on the **shutdown** command. Some examples of the **shutdown** command include:

- **shutdown -h now** shuts down the system with no time delay.
- **shutdown -h -t 90** shuts down the system in 90 seconds.
- **shutdown -r now** reboots the system with no time delay.



*Note: The **reboot** command essentially performs the same task as the **shutdown -r now** command.*

THE sleep COMMAND

The **sleep** command is used to pause system activities for a specified time. The command **sleep {seconds}** hangs up the prompt for the number of seconds specified.

SUPERUSER COMMANDS

In Linux, the user with administrator credentials is the superuser. The superuser is typically named root. It is generally a bad practice to log onto the system as the superuser, and you should get in the habit of logging in with a non-privileged account. However, there will be times when you need to assume the privileges of the superuser in order to perform an administrative task.

The **SU** – command ("substitute user") switches user credentials, and **SU – root** switches credentials to the root user. The system will prompt you to enter the root

user's password for authorization purposes. Once you are logged in as root, you will be able to perform tasks that you were previously unable to.

SHELL HISTORY

The Bash shell keeps a history file of all commands entered. You can reference this file and repeat commands, increasing efficiency and consistency of entered commands.

Examples of using shell history include:

- The **history** command outputs the most recently entered commands in a list format.
- The **Up Arrow** and **Down Arrow** keys cycle through the command history. Press **Enter** to reissue the desired command from the history.

TAB COMPLETION

The Bash shell supports **tab completion**, enabling users to type in enough of a command, file name, or directory name to be unique, then filling in the remainder of the entry. This feature reduces typographical errors and speeds up the entering of commands.

Examples of using tab completion include:

- Typing **his** and pressing **Tab** will automatically fill the rest of the **history** command.
- Typing **cd /home/user1/Aug** and pressing **Tab** will automatically fill the directory path to **cd /home/user1/AugustProjects** assuming such a directory already exists.

SHELL TIPS AND TRICKS

While the command-line interface can be intimidating, there are several ways of making it easier to work with and more efficient. As you get more comfortable with Bash, you may find you prefer working at the CLI much of the time.

Here are a few tips to help make working at the command-line easier:

- **Tab completion:** Get in the habit of using tab completion for speed and to minimize typos.
- **Use history instead of re-writing long commands:** When you make a typographical error in a command or file name, do not manually re-type the entire line. Repeat the line with the mistake by hitting the **Up Arrow** key one time, and then use the **Left** and **Right Arrow** keys to move to the mistake so that you can correct it.
- **Read the command backward:** When troubleshooting your commands, start from the right and read to the left. This method makes it a great deal easier to notice missing or duplicate characters.
- **Clear the screen:** Enter the **clear** command to clear the CLI of all text. This is useful when you're starting a new task and want to eliminate any distracting information from past command entries.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 1-3

Discussing Shell Commands

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **True or false? In Linux, Equipment.txt and equipment.txt refer to the same file.**

2. **Which command enables a user to log in to a system with their own credentials, and then to become the root user to perform administrative tasks?**

3. **A fellow administrator calls you and needs you to check the /etc/shadow file for a new user's entry, confirming that it indeed exists. How would you look at the contents of the /etc/shadow file without the possibility of changing the file?**

4. **You need to quickly create an empty file (log.txt) so that an application can copy data into it. Which command can you use to create the file?**

5. **You recall that a few days ago, you ran a command that helped you find a filtered list of files on the file system. You don't remember the command's somewhat complex syntax. How can you reproduce the command without searching the Internet for the correct syntax again?**

Activity 1-4

Entering Shell Commands

BEFORE YOU BEGIN

You have a CentOS 7 system that has already been installed for you, as well as a user account. The system is presenting you with a CLI.



Note: Activities may vary slightly if the software vendor has issued digital updates. Your instructor will notify you of any changes.

SCENARIO

As a result of your earlier discussion with the IT team at Develetech, your CTO is becoming more and more convinced of the viability of switching the company's server infrastructure to Linux. The CTO wants you to become more familiar with using Linux, and he suggests doing so by booting up a test machine and trying it out. So, you'll start by entering some basic commands at the Bash shell to get a feel for the Linux environment.



Note: Whenever the instruction states "enter <command>", you are required to type the command and press **Enter**. Note that Linux commands, command-line options, and file names are case-sensitive.



Note: Some of the commands in the course activity steps spill over onto the next line. Unless specified otherwise, all commands should be entered on the same line in the CLI.

1. Log in to the CLI.

- a) At the **localhost login** prompt, enter **Student##** where **##** is your student number.

For example, if your number is 03, you'd enter **student03**

- b) At the **Password** prompt, enter **Pa22w0rd**



Caution: Be careful when you type in passwords, as the CLI does not show them on screen.

- c) Verify that you are presented with a **[student##@localhost ~]\$** prompt, indicating that you have successfully logged in.

2. Enter your first command.

- a) At the prompt, enter **echo 'Hello, World!'**
- b) Verify that the console printed the string "Hello, World!" back to you.

3. Retrieve information about your current working directory.

- a) Enter **pwd**

- b) Verify that your current working directory is **/home/student##**
- c) Enter **ls** and note the contents of your current working directory.
At the moment, there doesn't seem to be anything in your home directory.
- d) Enter **ls -a** to display hidden items.
Now, you can see that there are files and folders in this directory, but they are hidden from a standard directory listing.
- e) Enter **ls -al** and note that you are given more information about each item in the directory.

```
drwxr-xr-x.  9 root      root          119 Dec 14 18:02 ..
-rw-----.  1 student01 student01    14877 Jan 10 21:54 .bash_history
-rw-r--r--.  1 student01 student01       18 Apr 11 2018 .bash_logout
-rw-r--r--.  1 student01 student01      208 Jan  2 15:47 .bash_profile
-rw-r--r--.  1 student01 student01      276 Jan  2 15:52 .bashrc
drwxrwxr-x. 18 student01 student01     4096 Jan 11 15:06 .cache
drwxrwxr-x. 22 student01 student01     4096 Jan 11 15:06 .config
drwx-----.  3 student01 student01      25 Dec 14 20:18 .dbus
drwxr-xr-x.  2 student01 student01      45 Jan  2 18:38 Desktop
```

The name of the file or folder is on the right. The last modified date and time is to the left of the name, and to the left of that is the size of the file or folder (in bytes). Most of the other fields relate to permissions and ownership.

4. Change your current working directory.
 - a) Enter **cd /etc**
 - b) Verify that the prompt has changed to **[student##@localhost etc]\$**
 - c) Enter **pwd** to further verify that your current working directory is now **/etc**
 - d) Enter **cd /var/log** to move to the directory where system log files are stored.
 - e) Enter **ls -al** to see all of the files and folders that exist in this directory.
 - f) Enter **cd /home/student##** where **##** refers to your student number to move back to your home directory.
5. Create a file in your home directory.
 - a) Enter **touch myfile** to create an empty file in your home directory.
 - b) Enter **ls** and verify that **myfile** is listed in the directory.
6. View some files using the **cat** command.
 - a) Enter **cat /etc/hostname**
 - b) Verify that the contents of the **/etc/hostname** file are printed at the CLI.
 - c) Enter **cat /var/log/dmesg**
 - d) Verify that this log file is so long that much of its contents scroll past the screen.
The **cat** command has no navigational options, so you'll need to use a more advanced command to view all of the log file.
7. View lengthy files using the **less** command.
 - a) Enter **less /var/log/dmesg**
Instead of scrolling, the **less** command printed only the beginning contents of the file that fit on the screen. Your prompt has also changed to the name of the file, highlighted.
 - b) Press **Enter** or **Down Arrow** to scroll down one line.
 - c) Scroll down a few more lines.
 - d) Press **y** or **Up Arrow** to scroll up one line.

- e) Scroll up a few more lines.
 - f) Press **Spacebar** or **Page Down** to scroll down an entire screen.
 - g) Press **b** or **Page Up** to scroll up an entire screen.
 - h) Press **q** to quit viewing the file and return to your regular prompt.
- 8.** Edit a text file with Vim.
- a) Enter **Vim myfile** to open the file you created earlier in the Vim text editor.
 - b) Press **i** to switch to insert mode.
 - c) Type **Hello, this is <your name>** and replace **<your name>** with your first name.
 - d) Press **Esc** to switch back to command mode.
 - e) Enter **:WQ** to save the file and quit.
 - f) Enter **cat myfile** and verify that the contents of the file are printed to the CLI.
- 9.** Create and edit a text file with GNU nano.
- a) Enter **nano myfile2** to create and begin editing a new file.
 - b) Type **Hello, this is <your name>** and replace **<your name>** with your first name.
 - c) Press **Ctrl+O**, then press **Enter** to save the file.
 - d) Press **Ctrl+X** to quit.
 - e) Enter **cat myfile2** and verify that the contents of the file are printed to the CLI.
 - f) Enter **clear** to clear the screen.
- 10.** Assume superuser privileges.
- a) Enter **cat /var/log/boot.log**
 - b) Verify that you are given a "Permission denied" error.
As a regular user, you do not have permission to read this log file. You need to elevate your privileges to do so.
 - c) Enter **SU - root**
 - d) At the **Password** prompt, enter **Pa22w0rd**
- 

Note: The passwords for your student account and the root account are the same for classroom convenience. In a production environment, the root password should be unique and not based on a common dictionary word.
- e) Verify that your prompt has changed to **[root@localhost ~]#**
You are now logged in as the root user, the user with the highest level of privileges (superuser).
 - f) Enter **cat /var/log/boot.log** and verify that you can now read the file.
 - g) Enter **exit** to log out as root and log back in to your regular student account.
- 11.** Leverage tab completion to make typing commands more efficient.
- a) Enter **touch thisisalongfilename.txt**
 - b) Type **ls -l th** and then press **Tab**.
 - c) Verify that the rest of the file name is filled at the command-line.
 - d) Press **Enter** to execute the command.
- 12.** View the command history.
- a) Type **his** and press **Tab**.
 - b) Verify that the **history** command is populated at the command-line. Tab completion works on file, directory, and even command names.

- c) Press **Enter** to execute the command.
 - d) Verify that a list of the commands you recently entered is printed on the screen.
- 13.** Restart the computer.
- a) Enter **reboot**
 - b) Verify that the computer restarts, and then prompts you to log in.
 - c) Log in as **student##** with **Pa22w0rd** as the password.
-

Topic C

Get Help with Linux

Now that you are familiar with the Linux shell, you may want to begin using commands in your system. However, there will be times when you need assistance with the various available commands. In this topic, you will identify the help and support options offered by Linux.

LINUX DOCUMENTATION

Documentation is a necessity in any major computing project, and Linux is no different. Documentation helps users of the Linux operating system, no matter their role or experience level, to perform a wide range of activities and resolve a wide range of issues. However, just like there is not one official form of Linux, neither is there a single, authoritative source of documentation. Documentation is spread across multiple sources that appear in multiple forms, each one suited to a particular context.

Some of major sources of Linux documentation include:

- Manual pages
- Built-in help commands
- Online documentation projects
- Usenet newsgroups
- Internet mailing lists
- Question and answer websites
- Forums and social media
- Books and other print resources

MANUAL PAGES

Linux **manual pages**, or man pages, contain the complete documentation that is specific to each Linux command. The man pages are available on Linux systems by default. The man page for a specific command is displayed using the **man** command. They usually include information such as the name of the command, its syntax, a description of its purpose, the options it supports, examples of common usage of the command, and a list of related commands.

Man pages are perhaps the most immediate source of help available when you need to learn more about what a command does or how to operate it. They are especially useful in situations where Internet access is not available. However, man pages can be difficult to parse for someone not familiar with how they're formatted.

Note: Man pages are presented in simple ASCII text format for ease of access.



```

MAN(1) ← Command → MAN(1)
root@localhost:~ Manual pager utils
MAN(1)

NAME
man - an interface to the on-line reference manuals

SYNOPSIS
man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m sys-
tem[,...]] [-M path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only]
[-a] [-u] [--no-subpages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation]
[--no-justification] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section]
page ...]
man -k [apropos options] regexp ...
man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
man -f [whatis options] page ...
man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager]
[-r prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z]
file ...
man -w|-W [-C file] [-d] [-D] page ...
man -c [-C file] [-d] [-D] page ...
man [-?V]

DESCRIPTION
man is the system's manual pager. Each page argument given to man is normally the name of
Manual page man(1) line 1 (press h for help or q to quit)

```

Information about the command

The man page for the man command.

SYNTA

X

The syntax of the **man** command is **man {command}**

SYNOPSIS FORMAT

Most of the components of a man page are self-explanatory, but the **SYNOPSIS** component can be somewhat confusing to new users. This part of a man page provides the syntax of the command along with some example use cases. These use cases are formatted as such:

- **bold** text should be typed exactly as shown.
- *italic* text should be replaced with the appropriate argument. Note that this may be formatted differently on certain systems, like underlined text or colored text.
- **[abc]** indicates that all arguments within the brackets are optional.
- **-a|-b** indicates that the arguments on either side of the pipe (|) cannot be used together.
- *italic* text with **...** (ellipsis) after it indicates that the argument can be repeated.
- **[italic]** text with **...** after it indicates that the entire expression within the brackets can be repeated.

man COMMAND OPTIONS

The **man** command supports different options. Some of the frequently used options are listed here.

Option	Description
-a	Finds all entries matching the query.
-D	Displays debugging information.
-f	Displays a short description of the command along with the man pages/sections.
-h	Displays help options for the man command.
-k	Lists all manual pages/sections containing the keyword along with their location.
-K	Searches for the specified string on all pages.

Option	Description
-t	Formats the man pages to enable printing.

MAN PAGE SECTIONS

Man pages for commands may be listed under one or more sections. A section defines what category the command belongs to. When a command has more than one section listed, it means that documentation for the same command is available from more than one source. These sections are identified by the number displayed beside the command; for example, **fsck (8)**

Various man page sections are provided in the following table.

Section Number	What It Contains
1	General commands
2	System calls
3	C library functions
4	Special files (usually found in /dev)
5	File formats and conventions
6	Games and screensavers
7	Miscellaneous
8	System administration commands and daemons

MAN PAGES NAVIGATION

You can navigate through Linux man pages using a number of keys. These keys are described in the following table.

Key	Used To
Home	Move to the beginning of the man page.
End	Move to the end of the man page.
Page Up	Scroll up the page progressively.
Page Down	Scroll down the page progressively.
/	Begin a search for a term or text string.
n	Move to the next occurrence of the search term.
p	Move to the previous occurrence of the search term.
q	Quit and return to the shell prompt.

OTHER BUILT-IN HELP OPTIONS

In addition to the **man** command, Linux offers other built-in options for help.

Help Option	Description
apropos	Sets the NAME section of all man pages for the keyword that the user provides. The NAME section usually contains a brief, one-sentence description of the command after the name itself. The apropos command is therefore useful when you'd like to perform some task, but don't know the name of the appropriate command(s). The syntax of this command is apropos {keyword}

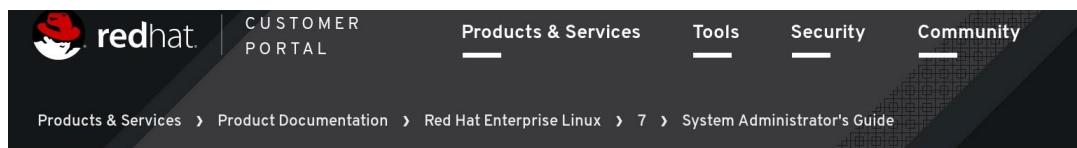
Help Option	Description
<code>whatis</code>	Displays the brief description of the given command, including commands that appear in multiple sections. This is essentially the opposite of <code>apropos</code> —you know the command name, but want to figure out what it does. The syntax of this command is <code>whatis {command}</code>
<code>info</code>	Displays the info page of a command. Info pages are essentially an alternative to man pages, and are favored by the GNU project. Although they have features such as hyperlinking and can easily generate HTML pages and PDF files, info pages are not as widely adopted as man pages. The syntax of this command is <code>info {command}</code>
<code>--help</code>	Displays a quick summary of the usage of a command and a list of arguments that can be used. This feature can be used with most commands in Linux. The syntax of this command is <code><command> --help</code> or <code><command> -h</code>
<code>/usr/share/doc/</code>	This directory contains documentation for libraries, system utilities, and other software packages installed on the system. Each software package has its own subfolder in this directory, and each subfolder usually contains separate documents that list the authors, licensing information, installation instructions, general README, etc.

ONLINE DOCUMENTATION

The Internet is one of the best places to go to for help with Linux. There is not one single online source that is necessarily better than the others; the choice often comes down to what type of help you're looking for. The following table lists some of the best online sources for Linux documentation.

Online Source	Description
Official distro documentation	Some of the more popular Linux distributions provide their users with documentation that applies specifically to that distro. This documentation often includes the release notes of different versions and updates, an installation guide, a virtualization guide, and more. For example, the official documentation for Ubuntu is available at https://help.ubuntu.com/ .
Linux Documentation Project (LDP)	The LDP is an all-volunteer project that provides a comprehensive resource for Linux documentation. The documentation comes in many forms, from HOWTOs that are step-by-step procedures for performing a specific task in Linux, to Guides, which include suggestions and best practices on a variety of Linux topics. The LDP also includes FAQs, man pages, and other types of documentation. The LDP is available at https://www.tldp.org/ .
Linux man-pages project	This is the official source of man pages for Linux kernel and C library interfaces that are available to user space programs, i.e., code that runs outside of the kernel. This project is available at https://www.kernel.org/doc/man-pages/ .

Online Source	Description
GNU coreutils manual	This is the official source of documentation for the GNU core utilities, or coreutils. The coreutils is the package of GNU software tools that is compiled with the Linux kernel, along with other software, to form a Linux distribution. This manual is available at https://www.gnu.org/software/coreutils/manual/ .



A screenshot of the Red Hat Customer Portal showing the 'System Administrator's Guide' for RHEL 7. The page title is 'SYSTEM ADMINISTRATOR'S GUIDE'. Below the title, it says 'RED HAT ENTERPRISE LINUX' and '7'. On the left, there is a sidebar with a search bar and a list of chapters from the guide, such as 'I. Basic System Configuration' and '1. Getting Started'. On the right, there is a 'Red Hat Training' section with a link to 'Red Hat Linux'.

The online documentation for the RHEL 7 distribution.

INTERACTIVE HELP

Online documentation is a quick and easy reference point, but it's not always the best source for answering your Linux questions, especially if those questions are complex or apply to unconventional scenarios. These questions are often best answered by interacting with other people directly, whether in real-time or asynchronously. The following table lists some of the major sources of interactive help on the Internet for Linux issues.

Online Source	Description
Usenet newsgroups	Usenet newsgroups are online discussion repositories similar to bulletin board systems, and are some of the oldest forms of Internet-based discussion. Users post text or files to a newsgroup, and other members of the group can view and respond to these posts, which are organized in a threaded manner. Newsgroups tend to be focused on specific subject matter; some newsgroups focused on providing Linux help include <code>comp.os.linux.help</code> , <code>comp.os.linux.answers</code> , and <code>comp.os.linux.admin</code> .

Online Source	Description
Mailing lists	Internet-based mailing lists are similar to newsgroups in that they involve threaded discussions among members of a specific community. "Posts" are essentially email messages sent to every user on a distribution list. You can find distro-specific mailing lists for several major distros, or you can subscribe to more general Linux mailing lists, such as those listed at https://lists.linuxfoundation.org/mailman/listinfo .
Q&A websites	Question and answer websites enable users to post a question on some topic, and other users can answer or otherwise comment on the question. Many of these sites have functionality for users to "like" or "upvote" questions they feel are useful, or answers they feel offer a good solution. The most popular Q&A site for IT topics is Stack Exchange, which has a Unix & Linux category available at https://unix.stackexchange.com/ .
Forums and social media	Internet forums are typically threaded posts, similar to newsgroups or mailing lists, but use web protocols to deliver the content. Some forums encourage users to post questions to the community, and other users respond to the posts with guidance. Aside from distro-specific forums, one popular forum for Linux support is available at https://www.linuxquestions.org/ . In addition, some social media sites, like Reddit, offer a forum-like discussion platform for soliciting help on specific topics—for example, https://www.reddit.com/r/linuxquestions/ .



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 1-5

Discussing Help in Linux

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **Since Linux is generally community (user) supported, name three sources of Linux documentation.**

 2. **To access complete local documentation, which command can you enter at the command-line to access extensive usage information on the cp (copy) command?**

 3. **A coworker sends you an instant message and asks you how to appropriately format a manual page for the mv command so that she can send it to the printer and get readable output. What command can she enter?**

 4. **Which online documentation site provides a comprehensive resource for Linux documentation including manual pages, FAQs, HOWTOs, and other types of documentation?**

 5. **If you know the name of a command and want to know what its function is, which command would you use to find out?**
-

Activity 1-6

Accessing Help in Linux

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

In order to be useful, Linux must have tools with certain capabilities that will be useful to the business. One of these capabilities is searching the contents of text files. This will come in handy for administrators who need to efficiently analyze text files like system logs, automated scripts, etc., for specific patterns. You'll need to find one or more Linux tools that can accomplish this and learn how to use them. So, you'll consult various help resources to get acquainted with the appropriate tool(s).

1. If necessary, log in as **student##** with a password of **Pa22w0rd**
2. Look for a command that could help you search the contents of a text file.
 - a) Enter **apropos search**
 - b) Verify that multiple commands are listed in the output, each of which includes the term "search" in its name or brief description.

You could try to pick out the appropriate command from these results, but changing your search might narrow them down.
 - c) Enter **clear** to clear the screen.
 - d) Enter **apropos pattern**
 - e) Verify that you receive fewer results.
3. **Looking at these results, which command(s) do you think would best fulfill the capabilities that you're looking for?**
4. Read the manual page for a command that could be what you're looking for.
 - a) Enter **man grep**
 - b) Verify that you see the manual page for the **grep** command.
 - c) Read the **SYNOPSIS** section to understand how to use the command.
 - d) Read the **DESCRIPTION** section to understand what the command does.
 - e) Navigate up and down the man page using the same keys as the **less** command.
 - f) Enter **/case** to search the man page for the term "case".
 - g) Press **n** to navigate to the next instance of the search term.
 - h) When you're at the end of the man page, press **Shift+N** to navigate to the previous instance of the search term.
 - i) Read the description for the command option that has to do with case.

5. Given what you've read in the man page for grep so far, answer what you think the following command does: `grep -i hello myfile`

 6. Search for more information about the grep command.
 - a) Press **q** to quit the man page.
 - b) Enter **cd /usr/share/doc**
 - c) Enter **ls**
 - d) Verify that there are many subdirectories in this directory, each of which corresponds to a software package.
 - e) Type **Cd grep** and press **Tab**.
-  *Caution: Make sure not to add a space after grep before you press Tab.*
- f) Verify that the path to the specific version of grep is completed, then press **Enter**.
 - g) Enter **ls**
 - h) Enter **less NEWS**
 - i) Briefly skim the change notes for the grep command.
 - j) Press **q** to quit.

7. How confident are you that this command fulfills what you're looking for?

 8. You still want to learn more about other commands that your team could use to search the contents of a text file. Aside from the help options built into Linux, what other sources can you consult in your research?
-

Summary

In this lesson, you identified the fundamental concepts behind the Linux operating system, as well as performed basic tasks to get you started in a Linux environment. This is the foundation on which you will build your skills in managing Linux systems.

How is Linux used in your organization? Are there any existing systems that could benefit from switching to Linux?

What is your preferred method for getting help with Linux, and why?

 **Practice Question:** Additional practice questions are available on course website.

Lesson 2

Managing Users and Groups

LESSON TIME: 2 HOURS, 30 MINUTES

LESSON INTRODUCTION

Now that you've performed some basic Linux® tasks, you're ready to start diving deeper into configuring the operating system for use by yourself and others. Before users can take advantage of Linux, you'll need to create and manage accounts for them. So, in this lesson, you will manage user and group accounts.

LESSON OBJECTIVES

In this lesson, you will:

- Assume superuser privileges when necessary.
- Create, modify, and delete user accounts.
- Create, modify, and delete group accounts.
- Query user and group accounts.
- Configure user account profiles.

Topic A

Assume Superuser Privileges



EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

In order to create accounts for other users, as well as perform many other administrative tasks discussed in this course, you'll need to ensure you have the privileges to do so. By assuming the privileges of the superuser, you'll be able to get your work done while still adhering to best practices for security.

USER ACCOUNTS

Accounts are objects that represent users and services to Linux. If these entities are represented by an account, then that account can be managed. **User accounts** represent identities that authenticate to the system and can use authentication credentials to do specific tasks. User information includes group memberships.

Individuals who will be using the Linux computer should have their own unique user accounts. Administrators will use these accounts to control the user's access to files, directories, and commands on the system. Each account is referenced by the system using a user ID (UID), rather than a name. Names are used for the convenience of the users.

User accounts have several attributes, including password information, group memberships, expiration dates, comments, etc.

TYPES OF USER ACCOUNTS

There are three different types of accounts: root, standard user, and service.

The root user account plays two roles on a Linux system. The first role is that of the local administrator. A user logged on as root can do administrative tasks such as password resets, system configuration changes, user account management, etc. The second role played by the root user account is to provide a security context for some applications and commands. These applications and commands may be called by the system or manually entered by a user logged on as root. The root user account in Linux is significantly more powerful than the local administrator account on a Windows® system. It is a bad practice to log on to any system with administrative credentials. On a Linux system, this can be particularly dangerous. The root user can take destructive action on the system, often without any verification prompt.



Note: Some Linux distributions, such as Ubuntu, disable the root user account entirely in order to minimize this threat. During the installation process, a regular user account is given administrative privileges.

Standard user accounts represent regular system users who may log on to run applications, configure databases, build websites, etc. Each user should have their own account and these accounts should not be shared. Most tasks that a user should be doing on the system should only require standard user account privileges. It is possible to set a standard user account to have administrative privileges. The advantage of this over permitting the user to log on as root directly is that the privileges of the standard user can be limited, whereas the root user privileges cannot.

Applications also consume resources on the system, so they are often represented by their own service accounts. These accounts are usually specific to the service (such as the **httpd** web service or a database service). They are disabled for regular log on, and the accounts are usually created as part of the service installation process. They will often own configuration files or executables associated with the service.

SUPERUSER

In Linux, the local administrator account is named root. The account is used to perform administrative functions like managing users, configuring devices, configuring network settings, etc. The system also runs services with root credentials. The system does not necessarily confirm with the root user potentially destructive commands. The authority to log on to the system with root credentials is usually associated with the knowledge of administrative functions.

The root account is also referred to as the **superuser**. The security best practice is to never log on to the system with administrative credentials, but rather to log on with a non-privileged account and elevate credentials when necessary.

PRINCIPLE OF LEAST PRIVILEGE

In information security, the **principle of least privilege** states that users should be given no more authority on the system than they need to perform their job. If a user needs to be able to read but not write to a file, then give them only read. If a user needs to be able to restart the server but not reconfigure the server, then only give them privileges to restart. It is much easier to correct privilege issues by giving a little more access than it is to remove existing access. By giving the user the access they need to do their jobs and no more than that, the system will remain significantly more secure.

THE su COMMAND

As you have seen, it is a poor practice to log on to the server directly with root privileges. The better practice is to log on with a standard user account, then elevate your privileges to root as needed. One way of elevating privileges is to "substitute user" using the **SU** command.

The **SU** command, without an option, enables a user to switch their identity to that of another user, but it retains the original user's profile and variables. The switched user also remains in the home directory of the original user. Anyone using **SU** except the root user will be challenged for the password of the user account they are switching to.

Using **SU** with a hyphen following it enables a user to change users and launch a new shell under the context of that new user. This is a much better practice. Anyone using **SU** – except the root user will be challenged for the password of the user they are switching to. It is most common to switch to the root user, but any user can switch to any other user so long as they know the user's password.

The screenshot shows a terminal window with the following text:

```
manderson@server01:~$ [student01@server01 ~]$ su - manderson
Password:
Last login: Mon Jan  7 13:34:08 GMT 2019 on pts/1
[manderson@server01 ~]$
```

Annotations on the screenshot:

- A blue callout bubble at the top left says "Logged in as initial user".
- A blue callout bubble at the bottom left says "Logged in as other user".
- A blue callout bubble at the top right says "Substitute other user". An arrow points from this bubble to the "->" part of the su command in the terminal.

Substituting another user for the current one.

SYNTAX

The syntax of the **SU** command is **SU [-] [user name]**



Note: Without a user name argument, the **SU** -command will assume you meant to sign in as root.

THE sudo COMMAND

With the **SU** command, any user who knows the root password can "get root" and do anything the root user can do. An account using **SU – ROOT** essentially is the server administrator. This is often much more power than should be delegated to users. A better practice is to delegate specific functions to users, rather than granting system-wide root privileges.

The **sudo** command enables the server administrator to delegate specific commands to specific users, without granting them full privileges on the server. Delegation is done in the **/etc/sudoers** file by using the **visudo** editor. Users and groups may be given specific commands to run in order to fulfill their responsibilities without having full administrator privileges.

SYNTAX

The syntax of the **sudo** command is **sudo [options] {command}**

THE sudoedit COMMAND

Some Linux files require root user privileges to edit. This could be accomplished with a **sudo** configuration, but a simpler and more secure option is to use the **sudoedit** command. This command permits a user to edit a file with their own credentials, even if the file is only available to the root user. In addition, the user can use their preferred text editor.

To use **sudoedit**, you must make an entry in the **sudoers** file. For example, the following line could be added to the **sudoers** file:

```
%editors ALL = sudoedit /path/to/file
```

Any member of the **editors** group could then enter the following command to edit a file: **sudoedit /path/to/file**

The **sudo** configuration is appropriate for commands that need to be executed with elevated privileges, while the **sudoedit** option is appropriate for files that need to be edited with elevated privileges.

SYNTAX

The syntax of the **sudoedit** command is **sudoedit [options] {file name}**

THE visudo COMMAND

While the **/etc/sudoers** file is a normal text file, it is essential not to directly edit it with a standard text editor like Vim or nano. The **/etc/sudoers** file controls access to all elevated privileges and a mistake in this file can render it impossible to gain root privileges on the server. Most distributions will set a default editor (usually Vim or nano) for **/etc/sudoers**. When using the **visudo** command, the system verifies the syntax of the **/etc/sudoers** file before committing changes, enabling the administrator an opportunity to correct mistakes before they become part of the running configuration.

```
/etc/sudoers file
root@server01:~#
## Allows members of the users group to shutdown this system
# %users localhost=/sbin/shutdown -h now

## Read drop-in files from /etc/sudoers.d (the # here does not mean a comment)
#includeinclude /etc/sudoers.d
student01 ALL=(ALL) NOPASSWD:ALL
```

/etc/sudoers
file

root@server01:~#

Entry granting
root privileges

The /etc/sudoers file with an entry granting root privileges to a user account.

SYNTA

X

The syntax of the **visudo** command is **visudo [options]**

visudo COMMAND OPTIONS

The following are some options you can use with the **visudo** command.

Option	Used To
-C	Check the existing sudoers file for errors.
-f {file name}	Edit or check a sudoers file in a different location than the default.

Option	Used To
-S	Check the sudoers file in strict mode—any aliases that are used before being defined will result in errors.
-X {file name}	Output the sudoers file to the specified file in JavaScript Object Notation (JSON) format.

THE wheel GROUP

The root system account is used for a great deal more than just administrative tasks. Many parts of the actual Linux operating system run under root credentials. Many distributions disable the actual root account for users and instead allow administrative functions based on membership in the **wheel** group.

Members of the **wheel** group exercise the administrative privileges of root with less potential for damaging the system. For example, members of the **wheel** group can use the **sudo** command to avoid having to sign in as the root user. You can use the **visudo** command to edit the privileges of the **wheel** group, if necessary. You can add users to the **wheel** group to give them privileges. Be very cautious about the membership of the **wheel** group.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 2-1

Discussing Superuser Privileges

SCENARIO

Answer the following questions to check your understanding of the topic.

-
1. **Name the three types of Linux user accounts.**

 2. **You need to add an administrator to the /etc/sudoers file to give them the ability to use the sudo command. Which command must you use to add users to the /etc/sudoers file?**

 3. **Why is it a security best practice to log onto a Linux system with a regular user account rather than with the root user account?**

 4. **Why is it important to put the principle of least privilege into practice?**

 5. **Describe the difference between using the su command and using the sudo command to perform administrative tasks.**
-

Activity 2-2

Assuming Superuser Privileges

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

While investigating Linux on behalf of Develetech, you have found multiple warnings about the danger of using the root user administrative account. You are already familiar with the principle of least privilege, which states that users should be granted only the level of access they need and no more. You also know that this applies to administrators as well as to end users. The Develetech security policy states that administrative privileges must be carefully controlled. You need to report on how this requirement can be satisfied.

1. If necessary, enter `cd ~` to return to your home directory.
2. Use the `su root` command to elevate your credentials to those of root.
 - a) Enter `id` to verify that you are currently signed in as `student##`
 - b) Enter `su root` to change to root.
 - c) Enter the `Pa22w0rd` password.
 - d) Enter `id` to verify the root user login.
 - e) Enter `pwd` to confirm the present working directory.
Note that while your credentials are those of the root user, your location and context are those of the `student##` user.
 - f) Enter `exit` to return to the `student##` user login.
3. Use the `SU - root` command to elevate your credentials and context to those of root.
 - a) Enter `SU - root` to change to root.

Caution: There is a space on each side of the hyphen.


 - b) Enter the `Pa22w0rd` password.
 - c) Enter `pwd` to confirm the present working directory.
Note that both your credentials and your context are those of the root user.

Note: If you use the `su` command without an argument, the system will default to the root user. Example: `su -` assumes `su - root`


4. Delegate administrative privileges to the student account.

- a) Enter **visudo** to start editing the **sudoers** file.
- b) Press **Page Down** several times to move the cursor to the bottom of the file.
- c) Press **End** to move to the end of the last line.
- d) Press **o** to enter insert mode and start a new blank line below the current line.
- e) Add the following text on a new line:

student## ALL=(ALL) NOPASSWD:ALL



Caution: Remember to replace **##** with your student number.

This grants the student account the ability to execute all commands without you having to switch to the root user every time. It also prevents you from having to input your password. This is for classroom convenience and is not suggested on a production environment.

- f) Press **Esc** to exit insert mode.
 - g) Enter **:WQ** to save and close the file.
5. Test your student account's ability to shut down the machine.

- a) Enter **exit** to return to your student account.
- b) Enter **id** to verify that you are signed in to your student account.
- c) Enter **exit** again to log out of the system.
- d) Log back in as **student##**
- e) Enter **sudo /sbin/shutdown -r 15**

This command tells the system to reboot after a fifteen minute delay. It requires administrative privileges. You are executing the command with **SUDO** in order to temporarily leverage those privileges.



Note: If you ever forget to add **sudo** to a privileged command, enter **sudo !!** to re-issue the most recent command with superuser privileges.

- f) Press **Ctrl+C**, and then enter the **sudo shutdown -C** command to interrupt the reboot.

Topic B

Create, Modify, and Delete Users



EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Now that you have superuser privileges, you're ready to begin creating accounts for users in your organization. You'll also, at some point, need to modify and delete those accounts as necessary.

THE useradd COMMAND

The `useradd` command is used to create user accounts and configure basic settings. As part of the account creation process, `useradd` references several files:

- The account is stored in the `/etc/passwd` file.
- The account is configured according to various options set in the `/etc/login.defs` file.
- The account's home directory is created at the `/home/<account name>` directory.
- The account's home directory is populated using files from the `/etc/skel` directory.

By default, the `useradd` command does not set a password for the account. Since most Linux distributions will not permit a blank password, the account will exist but is not yet usable.

SYNTAX

The syntax of the `useradd` command is `useradd [options] [user name]`

useradd COMMAND OPTIONS

The `useradd` command includes many options to customize user accounts, as detailed in the following table.

Option	Description	Example
<code>-c</code>	Sets the comment field, which is typically used as the field for the user's full name.	<code>useradd -c "User One" user1</code>
<code>-e</code>	Sets the account expiration date.	<code>useradd -e 2019/12/31</code>
<code>-s</code>	Sets the user's default shell.	<code>useradd -s /bin/ksh</code>
<code>-D</code>	View the default configurations for new users.	<code>useradd -D</code>

THE passwd COMMAND

The **passwd** command is used by root to set or reset a password for any user. A user can use the **passwd** command themselves to reset their own password. It is also used to set the initial password for a user after creating the account with the **useradd** command.



Note: The screen will not show any output when a user is setting a password. Some users will mistake this for a problem, but it is actually Linux hiding the number of characters being used in the password.

```
manderson@server01:~$ passwd
Changing password for user manderson.
Changing password for manderson.
(current) UNIX password: ← Verify current
password
New password: █ Enter desired new
password
```

A user changing their own password.

SYNTA

X

The syntax of the **passwd** command is **passwd [user name]** where **[user name]** can be used by root to set a specific user's password.

THE /etc/passwd FILE

The **/etc/passwd** file stores user account information. All accounts, default or user-specific, will be found in this file. It is common for administrators to reference this file to learn about specific user accounts on the system. Each account contains seven fields of information. Each field is separated by a colon. The fields are not necessarily all populated.

Field	Content
User name	The name the user logs into the system with.
Password	The password that is assigned to the user, usually represented as an X to indicate that the password is stored elsewhere.
User ID	The unique number that represents the user to the system.
Group ID	The unique number that indicates the user's primary group membership.
Comment	Typically represents the user's full name.
Home directory	The absolute path to the user's home directory.
Login shell	The path to the shell that is launched when the user logs in (usually /bin/bash).

EDITING THE /etc/passwd FILE

The proper way to edit the /etc/passwd file is via the `useradd`, `usermod`, and `userdel` commands. Manual editing of the file may result in mistakes that render the system unusable.

THE /etc/shadow FILE

The /etc/passwd file was once used to store the cryptographically hashed version of passwords. That file is world-readable, however, meaning that one user could see the hashed version of another user's password. By entering that hashed password in a password cracking utility, a user could discover another user's password.

The /etc/shadow file is the modern storage location for hashed passwords, as well as additional account information. This additional information includes password requirements and expiration information. Only root has access to the content of the /etc/shadow file, preventing users from attempting to crack each other's passwords.

```

root@server01:~#
root:$6$NvdhpQMRl0CfJPRq$M2KzXi8JRM1peoyTohFFvC.78
ln3qcqL5Bvgf4qg0VS1mkxE6wLMxy4hsIgEcqIPk160qf3iIt4
ShZp0/RkHo.:.:0:99999:7:;;
bin:*:17632
daemon:*:17632:0:99999:7:;;
adm:*:17632:0:99999:7:;;
lp:*:17632:0:99999:7:;;
sync:*:17632:0:99999:7:;;
shutdown:*:17632:0:99999:7:;;
halt:*:17632:0:99999:7:;;
mail:*:17632:0:99999:7:;;
operator:*:17632:0:99999:7:;;

```

The screenshot shows the contents of the /etc/shadow file. A callout labeled "Root user entry" points to the first line. Another callout labeled "Hash data" points to the hashed password value for the root user. A callout labeled "Account info" points to the account information fields (UID, GID, and other parameters) for the root user.

The /etc/shadow file.

THE /etc/shadow FILE FORMAT

The following table details the format of the /etc/shadow file.

Field	Content/Additional
User name	The name the user logs into the system with.
Password	The hash value of the password that is assigned to the user.
Days since password changed	Days are counted from January 1, 1970.
Days before password must be changed	Typically set as 1 day.
Days until user is warned to change password	A value of 99999 means the password never needs to be changed.
Days after password expires that account is disabled	Ideally, this should be immediate.

Field	Content/Additional
Days the account has been disabled	Days are counted from January 1, 1970.
Unused field	Reserved for potential use in the future.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 2-3

Creating User Accounts

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

Managing user and group accounts in Linux will be a key administrative responsibility at Develetech. Now that you have become comfortable with some basic Linux commands, you need to become proficient at managing users. You'll start by creating some user accounts and viewing their defaults.

1. View the current default settings for new users.
 - a) Enter **sudo useradd -D** to view the default settings for newly created users.
 - b) Enter **less /etc/login.defs** to view the default settings for newly created users.
 - c) Press **q** to quit.
 - d) Enter **ls -a /etc/skel** to view files that will be copied to the home directories of newly created user accounts.

2. Create a user account for Michael Anderson named **manderson**.
 - a) Enter **sudo useradd manderson** to create a new user.
 - b) Enter **cat /etc/passwd** to view the new user account in the **/etc/passwd** file.

```
tcpdump:x:72:72::sbin/nologin
student01:x:1000:1000:student01:/home/student01:/bin/bash
manderson:x:1001:1001::/home/manderson:/bin/bash
```

Note: Newly created user accounts are appended to the bottom of this file.



3. Create a new user account for Chris Mason named **cmason**.
 - a) Enter **sudo useradd -c "Chris Mason" cmason**

This creates the **cmason** account and populates the comments field of the account with the user's full name.

 - b) Enter **cat /etc/passwd**
 - c) Verify that the newly created user account at the bottom of the screen also includes a "comment" consisting of the user's full name.

4. Create new user accounts for Andrew Riley and Rachel Alexander named **ariley** and **ralexander**, respectively.
 - a) Enter **sudo useradd ariley**
 - b) Enter **sudo useradd ralexander**

5. Create a new temporary user account for Rose Stanley named **rstanley** whose contract will end on December 31, 2025.
 - a) Enter **sudo useradd -e 2025/12/31 rstanley**
 - b) Enter **cat /etc/passwd** and note the newly created account.

THE chage COMMAND

The **chage** or "change age" command is used to control password expiration, expiration warnings, inactive days, and other information for existing accounts. Changes to a security policy or potential security breach may prompt the administrator to alter the password settings for existing accounts. These changes are reflected in the **/etc/shadow** file.

Option	Description	Example
-E	Sets the account to expire at the specified date and time.	chage -E 2022/12/31 user1
-I	Lists password aging information.	chage -I user1
-M	Sets the maximum days the password is valid for.	chage -M 90 user1
-m	Sets the minimum days until the password can be changed.	chage -m 1 user1
-W	Sets number of days before expiration that user will be warned to change their password.	chage -W 5 user1

SYNTAX

The syntax of the **chage** command is **chage [options] {user name}**

THE usermod COMMAND

The **usermod** command is used to modify settings for regular users. It edits the **/etc/passwd** file, avoiding the need for administrators to edit the file directly. There are many modifications an administrator can make to an existing user account.

The following table lists some options for the **usermod** command.

Option	Description	Example
-c	Sets the comment field.	usermod -c "User One" user1
-e	Sets the account expiration date.	usermod -e 2020/12/31 user1
-aG	Adds user to a group and keeps them in their existing groups.	usermod -aG sales-group user1
-l	Changes the user's login name.	usermod -l user99 user1

SYNTAX

The syntax of the usermod command is `usermod [options] {user name}`

LOCK USER LOGIN

An administrator may lock a user account if that user leaves the company, if there's a security breach, or if the user takes a long leave of absence. Locking the account renders it unusable without deleting the account or its settings. The account can be unlocked when needed.

User accounts can be locked with either the `passwd` or `usermod` commands. To lock:

- `passwd -l {user name}`
- `usermod -L {user name}`

To unlock:

- `passwd -u {user name}`
- `usermod -U {user name}`

Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.



Activity 2-4

Modifying User Accounts

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created several new user accounts.

SCENARIO

Now that you have configured a few standard user accounts, you want to ensure the accounts exist. You also need to set password requirements. In addition, you will investigate whether password expirations can be configured and whether user accounts can be locked if users take a leave of absence.

1. Display the contents of the /etc/passwd file.

- a) Enter **cat /etc/passwd**
- b) Verify that, for each user account, the password field shows an X character.

```
student01:x:1000:1000:student01:/home/student01:/bin/bash
manderson:x:1001:1001::/home/manderson:/bin/bash
cmason:x:1002:1002:Chris Mason:/home/cmason:/bin/bash
ariley:x:1003:1003::/home/ariley:/bin/bash
ralexander:x:1004:1004::/home/ralexander:/bin/bash
rstanley:x:1005:1005::/home/rstanley:/bin/bash
```

This indicates that the password hash is actually stored elsewhere.

2. Display the contents of the /etc/shadow file.

- a) Enter **sudo cat /etc/shadow**
- b) Verify that you can see various information about each user account, including their password hash value and any expiration information.

```
student01:$6$1wG1kxRk4xTphJPb$aEC.JApWLFn3HkuAtUazQykV0PPR6u0DG1sjQGSAR9
bejUQqL5VTrZAAo15du0xy5GevqplvvpGZf90iFHC1.:!:0:99999:7:::
manderson:!!:17879:0:99999:7:::
cmason:!!:17879:0:99999:7:::
ariley:!!:17879:0:99999:7:::
ralexander:!!:17879:0:99999:7:::
rstanley:!!:17879:0:99999:7:::20453:
```



Note: The !! symbols indicate that the account has a blank password and that users are not allowed to log in as that account.

3. Configure passwords for the user accounts.

- a) Enter **sudo passwd manderson**

- b) When prompted for the password, enter **Pa22w0rd**



Note: You can ignore the warning about this password failing a dictionary check. In a production environment, you'd choose a much stronger password.

- c) When prompted to retype the password, enter **Pa22w0rd** again.
 d) Repeat these steps to add the password for **cmason**, **rstanley**, **ariley**, and **ralexander**
 e) Enter **sudo cat /etc/shadow** and note that the password hash fields are now populated for these users.

4. Attach a real name to each user account.

- a) Enter **sudo usermod -c "Rose Stanley" rstanley** to modify the existing **rstanley** account.
 b) Repeat the previous step for each user account:
 - **manderson** — Michael Anderson
 - **ariley** — Andrew Riley
 - **ralexander** — Rachel Alexander
 c) Enter **cat /etc/passwd** to view the modifications.

5. Configure account expiration information.

- a) Enter **sudo chage -l manderson** to list the **manderson** account password information.
 b) Enter **sudo chage -E 2026/12/31 manderson** to set the account expiration for the user to 12/31/2026.
 c) Enter **sudo chage -l manderson** to view the new expiration information.

6. Configure user account lockouts.

- a) Enter **sudo passwd -l cmason** to lock the **cmason** account.
 b) Enter **sudo passwd -u cmason** to unlock the **cmason** account. Note the warning message.
 c) Enter **sudo usermod -L cmason** to lock the **cmason** account.
 d) Enter **sudo usermod -U cmason** to unlock the **cmason** account.

THE **userdel** COMMAND

The **userdel** command is used to delete user accounts. By default, it does not delete the user's home directory, unless you use the **-r** option. Deleting the user account removes all references to it. You would have to recreate the account and re-add it to groups in order to resemble the original identity. Use caution before deleting a user account.

SYNTAX

The syntax of the **userdel** command is **userdel [options] {user names}**



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 2-5

Discussing Creating, Modifying, and Deleting Users

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **In which file are user accounts stored?**

2. **What are the three commands used to properly edit the /etc/passwd file?**

3. **In which file are hashed passwords stored?**

4. **Why are /etc/passwd and /etc/shadow different files?**

5. **With which command can you change the default user shell to the KornShell for user bsmith?**

Activity 2-6

Deleting a User Account

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created the **ralexander** account.

SCENARIO

You recognize that part of the user account lifecycle is the deletion of accounts that are no longer needed on the system. You will use the **userdel** command to delete a test account.

Delete Rachel Alexander's account.

- a) Enter **cat /etc/passwd** and confirm the **ralexander** account exists.
- b) Enter **Sudo userdel ralexander** to delete the **ralexander** account.
- c) Enter **cat /etc/passwd** and confirm the **ralexander** account has been deleted.
- d) Enter **ls /home** and observe that the **ralexander** home directory still exists.

```
[student01@localhost ~]$ ls /home  
ariley cmason manderson [ralexander] rstanley student01
```

Topic C

Create, Modify, and Delete Groups



EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

In order to grant access to users in a more efficient manner, you'll create groups for users to be part of. You'll also modify and delete those groups as necessary.

GROUP ACCOUNTS

Groups associate user accounts that have similar security requirements. Groups simplify administrative tasks, allowing multiple users to be granted permissions to resources. Groups are represented on the system by a group ID number (GID). Users may be a member of more than one group.

THE /etc/group FILE

The `/etc/group` file is the storage location for groups. All groups, default or user-specific, will be found in this file. It is common for administrators to reference the `/etc/group` file to find information about the groups on the system. Each group contains four fields of information. Each field is separated by a colon. The fields are not necessarily all populated.

```
root@server01:~  
stapsys:x:157:  
stapdev:x:158:  
student01:x:1000:student01  
manderson:x:1001:  
cmason:x:1002:  
ariley:x:1003:  
rstanley:x:1005:  
MarketingDept:x:1008:ariley  
FinanceDept:x:1009:manderson  
GraphicsDept:x:1006:rstanley,jrobinson  
jrobinson:x:1010:  
apache:x:48:
```

Groups

Users in groups

The `/etc/group` file.

The proper way to edit the `/etc/group` file is with the `groupadd`, `groupmod`, and `groupdel` commands. Manually editing the file is not recommended, as a mistake could render the system unusable.

Field	Description
Group name	The user-friendly name of the group.
Password	The password required to enter the group.
Group ID	The group ID by which the system references the group.
Group list	The list of all group members (empty by default).

THE groupadd COMMAND

The `groupadd` command creates a group. By default, the group has no members and no password. In addition to creating a group with a friendly name, you can also specify a group ID using the `-g` option.

Some `groupadd` options include the following.

Option	Used To	Example
<code>-g</code>	Assign a group ID.	<code>groupadd -g 123 sales</code>
<code>-f</code>	Exit with a success status if the group already exists.	<code>groupadd -f sales</code>
<code>-o</code>	Allow a group to be created with a non-unique group ID.	<code>groupadd -o -g 123 sales</code>

SYNTAX

The syntax of the `groupadd` command is `groupadd [options] {group names}`

THE groupmod COMMAND

The `groupmod` command is used to change the group's own attributes. It will edit the `/etc/group` file for you. Modifications of the group might include changing its name or GID.



Note: Adding a user to a group is considered to be a modification of the user, not the group. As such, it is accomplished using the `usermod` command.

Some `groupmod` options include the following.

Option	Used To	Example
<code>-g</code>	Change the group ID.	<code>groupmod -g 123 sales</code>
<code>-n</code>	Rename a group.	<code>groupmod -n newsales sales</code>

SYNTAX

The syntax of the `groupmod` command is `groupmod [options] {group names}`

THE groupdel COMMAND

The `groupdel` command will delete groups from the `/etc/group` file. It does not delete user accounts that are members of the group. Exercise caution when deleting groups as a mistake can cause users to not be able to access resources.

SYNTAX

The syntax of the `groupdel` command is `groupdel [options] {group names}`



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 2-7

Discussing Creating, Modifying, and Deleting Groups

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **Why do administrators, classes, and best practices state that you should manage users by managing groups?**

2. **Which file contains the groups and user members of those groups?**

3. **Which three commands does a system administrator use to properly edit a Linux system's group file?**

4. **Which command does a system administrator use to add a user (bsmith) to an existing group (finance)?**

5. **True or false? The groupmod command is used to change the name of an existing group.**

Activity 2-8

Creating, Modifying, and Deleting Groups

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created several user accounts.

SCENARIO

You will need to associate several user accounts together into groups to make IT management at Develetech easier. You will create several groups that correspond to different departments. At some point, you'll need to rename the Graphics group to fit the naming scheme of the other groups. In addition, you will add users to the groups.

Part of the user/group management lifecycle dictates that you'll occasionally need to delete groups. So, you'll finish by deleting a group, but not the users that are part of that group.

1. Create a new group called **Graphics**.
 - a) Enter **cat /etc/group** to view the current groups on the system.
 - b) Enter **sudo groupadd Graphics** to create a new group called "Graphics".
 - c) Repeat this step to create three additional groups with the following names:
 - **SalesDept**
 - **MarketingDept**
 - **FinanceDept**
 - d) Enter **cat /etc/group** and note the presence of the four new groups.

Graphics:x:1006:
SalesDept:x:1007:
MarketingDept:x:1008:
FinanceDept:x:1009:

2. Rename a group with the **groupmod** command.
 - a) Observe the current **Graphics** group name.
 - b) Enter **sudo groupmod -n GraphicsDept Graphics** to rename the **Graphics** group to **GraphicsDept**

- c) Enter **cat /etc/group** and view the new group name.
3. Add users to groups with the **usermod** command.
- a) Enter **Sudo usermod -aG GraphicsDept rstanley** to add the **rstanley** account to the **GraphicsDept** group.
 - b) Repeat this step to add the following users to the following groups:
 - **FinanceDept** —**manderson**
 - **SalesDept** —**cmason**
 - **MarketingDept** —**ariley**
 - c) Enter **cat /etc/group** and confirm that each user is a member of their assigned group.
4. Delete a group with the **groupdel** command.
- a) Confirm that the **SalesDept** group exists.
 - b) Enter **sudo groupdel SalesDept** to delete the **SalesDept** group.
5. Verify that you deleted the group, but not its users.
- a) Enter **Cat /etc/group** to view the existing groups.
 - b) Confirm that the **SalesDept** group has been deleted.
 - c) Enter **cat /etc/passwd** to view the existing users.
 - d) Confirm that deleting the **SalesDept** group did not delete the **cmason** user account, even though it was a member of that group.
-

Topic D

Query Users and Groups



EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

After you've created and managed users and groups, you'll likely need to learn more about one or more accounts and their activity on the system. By querying users and groups, you can keep better track of how these accounts are being used.

ACCOUNT QUERYING

Administrators and users may need to gather information about their identity on the system. There are many commands that will report user and group information. This information is useful for troubleshooting access problems or verifying what account the user is currently logged on with.

THE whoami COMMAND

The `whoami` command is used to display the user name with which you are currently logged in to the system. Sometimes, you may need to log in to a system and switch among different users, and you may not be sure with which user you are currently logged in. In such instances, you can use the `whoami` command to verify your current user name.

```
root@server01:~ [root@server01 ~]# whoami
root
[root@server01 ~]#
```

User currently logged in

Displaying the user currently logged in.

COMMAND PROMPT IDENTIFICATION

Many Linux distributions will show the user name of the currently logged in user at the command prompt. For the root user, the prompt will show a # character. For standard users, the prompt will show a \$ character.

THE who COMMAND

The **who** command is used to determine the details of users currently logged in to a system. The output of the **who** command includes the user name, the name of the system from which the user is connected, and the date and time that the user has been connected since.

```
[root@server01 ~]# who
student01 :0          2019-01-02
                    18:41 (:0)
student01 pts/1        2019-01-02
                    19:07 (:0)
[root@server01 ~]#
```

Showing the details of users currently logged in.

SYNTAX

X

The syntax of the **who** command is **who [options]**

who COMMAND OPTIONS

The **-U** option can be used to see how long users have been idle. A dot indicates that the users were active up to the last minute, **old** indicates that the users have been inactive for over 24 hours, and anything between 2 minutes and 23 hours 59 minutes shows the length of time they have been idle. The **am i** option displays information only for the user who runs the command.

THE w COMMAND

The **w** command is primarily used to display the details of users who are currently logged in to a system and their transactions. The first line of the output displays the status of the system. The second line of the output displays a table with the first column listing the users logged in to the system and the last column indicating the current activities of the users. The remaining columns of the table show different attributes associated with the users.

```
[root@server01 ~]# w
14:27:05 up 4 days, 19:46, 2 users, load average: 0.05, 0.04, 0.17
USER   TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
student0 :0           :0          Wed18    ?xdm?   11:07m  1.37s /usr/lib
student0 pts/1        :0          Wed19     1.00s   0.63s  6.28s /usr/lib
```

Displaying user details and transactions.

SYNTAX

The syntax of the **w** command is **w [options] [user name]**

THE last COMMAND

The **last** command displays the history of user login and logout actions, along with the actual time and date. It also has options that enable you to filter users who have logged in through a specific terminal. For example, **last 1** will display the details of users who logged in using the first terminal. The **last** command retrieves information from the **/var/log/wtmp** file.

```
[root@server01 ~]# last
student0 pts/1      :0
student0 pts/0      :0
student0 pts/0      :0
reboot             .0-86
student0 pts/0      :0
student0 :0         :0
student0 pts/0      :0
student0 :0         :0
Wed Jan 2 18:44 - 13:30 (4+18:45) in
18:41              18:40
```

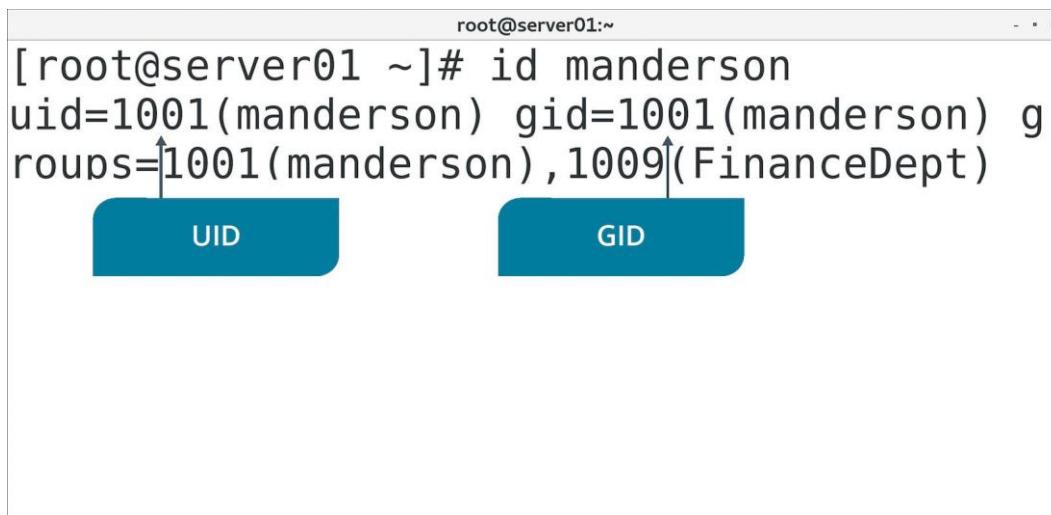
Displaying user login/logout history.

SYNTAX

The syntax of the **last** command is **last [options]**

THE id COMMAND

The **id** command is used to display user ID (UID) and group ID (GID) information. Entering the command with no options displays information about the user who is currently logged in. You can also specify a user name as an option to display ID information about a specific user.



```
root@server01 ~]# id manderson
uid=1001(manderson) gid=1001(manderson) g
roups=1001(manderson),1009(FinanceDept)
```

Showing UID and GID information for a specific user.

SYNTA

X

The syntax of the **id** command is **id [options] [user name]**

 **Note:** To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 2-9

Discussing Querying Users and Groups

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **As a Linux administrator, you might often switch accounts from yours to the root user or to other user accounts to perform tasks. Which command can you issue to find out which user account you're currently using?**

 2. **You suspect that a hardware failure is imminent and you need to reboot the system to ensure that everything is working properly or to force a failure. Before issuing the reboot command, which command can you use to check to see if other users are logged onto the system?**

 3. **A user reports that she was working on an important script when the system rebooted at approximately 6:30 P.M. last night. No warning was given. How can you find out who was logged into the system at that time and who could have rebooted the system without warning?**

 4. **As a system administrator, why might you issue the id command?**

 5. **As an administrator, you might need to reboot a system or otherwise perform maintenance. Which command would you issue to not only view logged on users but also their current activity?**
-

Activity 2-10

Querying Users and Groups

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

There are several ways a user can gather information about their own account and group memberships. In addition, there are multiple ways of identifying what users might currently be logged on the system. You will explore these methods to ensure you can answer questions the users you support might have. The Develetech security policy requires that a log file of user logins be kept in case of an audit or security incident.

1. Log in as the root user and verify your credentials.
 - a) Enter **SU - root** and the password to switch to the root user.
 - b) Enter **whoami** to see your login name.
 - c) Enter **id** to see your login credentials and group membership.
 - d) Verify that the command prompt shows the **root** name and a **#** character. This character also indicates that you are signed in as the root user. For standard users, the prompt will show a **\$** character.
 - e) Enter **exit** to leave the root login and return to your student account.
2. Verify your student account credentials.
 - a) Enter **whoami** to see your login name.
 - b) Enter **id** to see your login credentials and group membership.
 - c) Verify that the command prompt shows the **student##** name and a **\$** icon.
3. Check for information about users that are or have been logged in to the system.
 - a) Enter **who** to see what users are currently logged in to the system.
 - b) Enter **w** to see what users are currently logged in.
 - c) Compare **who** and **w** for details, and then observe the idle time information.
 - d) Enter **last** to see a record of recent logins to the system.

Topic E

Configure Account Profiles



EXAM OBJECTIVES COVERED

2.2 Given a scenario, manage users and groups.

In many cases, individual users will have their own preferences or unique needs when it comes to working in a Linux environment. So, you'll configure users' profiles to ensure they each have customized environments to work in.

USER PROFILES

Individual users have personal habits and preferences for their Linux work environment, and so a variety of profile file configurations are available to help them customize their experience. These customizations are referred to as profiles. An initial profile is created when the user account is created. There are many files and directories from which information is initially pulled to create a standard profile.

THE .bashrc FILE

The `.bashrc` file enables customization of the user's own environment. The file is stored in a user's home directory. Because the `.bashrc` file is unique to each user, it can be configured to a user's own specific needs or preferences.

A good example of the use of the `.bashrc` file is the use of aliases. Users can specify their own abbreviated commands without impacting the experience of any other user on the system. Another common configuration within `.bashrc` is environment variables. Users can also use the file to customize the command prompt to provide the information they want.

```
root@server01:~#
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
```

Alias definitions

The `.bashrc` file.



Note: The "dot" in front of the file name makes the file hidden. This is not a security configuration, but rather makes the user's home folder appear less cluttered.

EXAMPLE CONFIGURATIONS

Other than creating aliases, the `.bashrc` file is often configured to set default directory and file permissions for the user. Also, the default command prompt can be altered to provide more information. Most distributions place the user name, system hostname, and current directory in the prompt, but that can be changed.

THE `.bash_profile` FILE

The `.bash_profile` file provides shell configuration for the initial login environment. This differs from the `.bashrc` file, which provides settings for all of the user's interactive shells. The `.bash_profile` file is only read with the first login, while the `.bashrc` is read with all subsequent logins.

A default `.bash_profile` can be provided to new users via the `/etc/skel` directory.

```
root@server01:~#
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin } Shell environment
export PATH configuration
```

The `.bash_profile` file.

GLOBAL USER PROFILES

An administrator may find it desirable to define system-wide settings or to configure initial settings for users. The files and directories that follow give the administrator the flexibility to make a variety of configurations, which may be later customized by a user for their own specific needs.

THE `/etc/skel/` DIRECTORY

The contents of the `/etc/skel/` directory are automatically copied into the home directories of new users. Administrators can pre-populate the `/etc/skel/` directory with configuration files or other content. When the `useradd` command is run, the `/etc/skel/` directory's contents are copied to the new user's home directory, immediately giving them the configurations they might need.

Note that files added to the `/etc/skel/` directory after a user account is created will not be copied to the home directories of existing users.

THE /etc/profile FILE

The **/etc/profile** file provides system-wide environment variables. This may be more effective for administrators to configure if there are settings that apply to all users.

During the initial login process for a user, the system reads the **/etc/profile** file first for Bash shell configurations, and then any user-specific Bash customizations are pulled from the **.profile** file located in the user's home directory. The **.profile** file runs each time a new shell is started, whereas **/etc/profile** is only run at login. This approach enables administrators to define global shell settings, but still allow user-specific customizations.



Note: The global file is */etc/profile* (without the "dot" as the first character of the file name), while the user-specific file is *.profile*, indicating that the file is hidden.

EXAMPLE

An example of a **.profile** is as follows:

```
PATH=$PATH:$HOME/bin:/scripts
MAIL=/var/mail/$LOGNAME
EDITOR=emacs
export PATH MAIL EDITOR
```

The first line defines the paths of executable files; the second line defines the path for where incoming email messages are stored; and the third line defines the default text editor. The last line actually ensures these variables are implemented in the environment.

THE /etc/profile.d/ DIRECTORY

The **/etc/profile.d/** directory serves as a storage location for scripts administrators may use to set additional system-wide variables. It is recommended you set the environment variables via scripts contained in **/etc/profile.d/** rather than editing the **/etc/profile** file directly.

THE /etc/bashrc FILE

The **/etc/bashrc** file provides system-wide Bash settings. This is a little different than **/etc/profile**, which is used for variables.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 2-11

Discussing Account Profiles

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **True or false? Using a dot at the beginning of a file name makes the file more secure.**
2. **What is the primary difference between the .bashrc and the .bash_profile files?**
3. **Which directory's contents are copied to the user's home directory upon account creation?**
4. **Which file forces system-wide customizations for all users on a system that a user cannot change?**
5. **Where should administrators set system-wide variables on a Linux system rather than editing the /etc/profile file directly?**

Activity 2-12

Configuring Account Profiles

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You're concerned that a change to Linux systems may be difficult for users. You need to identify what files can be used to make the user command-line environments customized and consistent. In addition, you need to place a copy of the Develetech policies in each new user's home directory for reference.

1. View the user accounts' `.bashrc` files.
 - a) Enter `cat .bashrc` to view the configuration file for the student account. Notice that there are no preconfigured alias settings for standard users in CentOS 7.
 - b) Enter `sudo cat /root/.bashrc` to view the configuration file for the root user. Notice that the root user's profile includes alias settings for the copy, move, and delete commands, setting them for interactive mode. These are default alias settings for the root user in CentOS 7.
2. View the contents of the `.bash_profile` file.
 - a) Enter `cat .bash_profile` to view the contents of the configuration file. The `.bash_profile` file is called when the user first logs in. Observe that the file contains the `PATH` variable setting, which defines where Bash will search for command executables.
3. Add a file to the `/etc/skel` directory, create a new user, and then verify that the new file was copied to the new user's home directory.
 - a) Enter `ls -a /etc/skel` to view the files currently in this directory.
 - b) Enter `sudo touch /etc/skel/policies.txt` to create a file in the directory.
 - c) Enter `sudo useradd jrobinson` to create a new user account for Jerry Robinson.
 - d) Enter `sudo ls -a /home/jrobinson` and note the presence of the `policies.txt` file.
This file was copied as part of the `useradd` tool.
4. Configure the `jrobinson` user account.
 - a) Enter `sudo usermod -aG GraphicsDept jrobinson` to add `jrobinson` to the `GraphicsDept` group.

- b) Enter **sudo usermod -c "Jerry Robinson" jrobinson** to provide a full name in the comments field.
 - c) Enter **Sudo passwd jrobinson** to set a password for the account.
 - d) Enter **Pa22w0rd** as the password.
-

Summary

In this lesson, you created and managed user and group accounts. This will help you efficiently organize and maintain a Linux environment with numerous users.

How might you design groups in your organization?

How might you manage root privileges in your organization?



Practice Question: Additional practice questions are available on the course website.

Lesson 3

Managing Permissions and Ownership

LESSON TIME: 2 HOURS

LESSON INTRODUCTION

Creating accounts is more than just about allowing people to log in; it's also necessary for delegating access to system resources. However, managing accounts is not enough to accomplish this. The other important part of the process is configuring permissions and ownership. In this lesson, you'll ensure that the right people have the right access to the right resources in Linux®, while ensuring that no one has more access than is necessary.

LESSON OBJECTIVES

In this lesson, you will:

- Modify permissions for files and directories.
- Modify ownership of files and directories.
- Configure special permissions and attributes.
- Troubleshoot issues with permissions and ownership.

Topic A

Modify File and Directory Permissions



EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

You'll begin restricting access to files and directories by applying the proper permissions to those resources. This is a crucial step in ensuring the security of your data.

PERMISSIONS

Permissions are access rights assigned to users, which enable them to access or modify certain files and directories. Permissions can be set at different levels and for different access contexts. They enable you to configure who is allowed to access an object, and who is restricted from accessing that object. Controlling access through permissions mitigates risk by ensuring that users are only able to access what they need to get their job done, and no more.

THE **ls -l** COMMAND

The **ls -l** command gives you a long list of the files and directories in your current working directory. Each item in the list contains seven columns, three of which display permission information. The contents of the columns are described in the following table.

Column Number	Description
1	The permission string. This identifies if the item is a file or directory, the user, group, and other permission assignment, and the access method.
2	Number of links. Files generally have a link count of 1. For directories, the link count is the number of directories under it plus 2; 1 for the directory itself and 1 for the parent. Links are similar to Windows® shortcuts; they point to the location where the file exists and enable you to access and view the file.
3	Displays the owner of the file or directory.
4	Displays the group that has been granted access by the administrator. All members of this group have the group permission listed in the permission string. The administrator adds users to a group so that permissions can be assigned to the group instead of to each user.
5	Lists the size (in bytes) of the file or directory.
6	Displays the date and time the file was created or last modified.
7	Displays the file or directory name.

PERMISSION ATTRIBUTES

Permission attributes define exactly what a user is allowed to do with a particular file or directory. The following table describes the three permission attributes. Note that these attributes behave differently based on whether they apply to a file or a directory.

Attribut	Description
Read (r)	<ul style="list-style-type: none"> Files: The ability to access and view the contents of a file. Directories: The ability to list the contents of a directory.
Write (w)	<ul style="list-style-type: none"> Files: The ability to save changes to a file. Directories: The ability to create, rename, and delete files in a directory. Requires the execute attribute to also be set.
Execute (x)	<ul style="list-style-type: none"> Files: The ability to run a script, program, or other software file. Directories: The ability to access a directory, execute a file from that directory, or perform a task on that directory (e.g., a search).

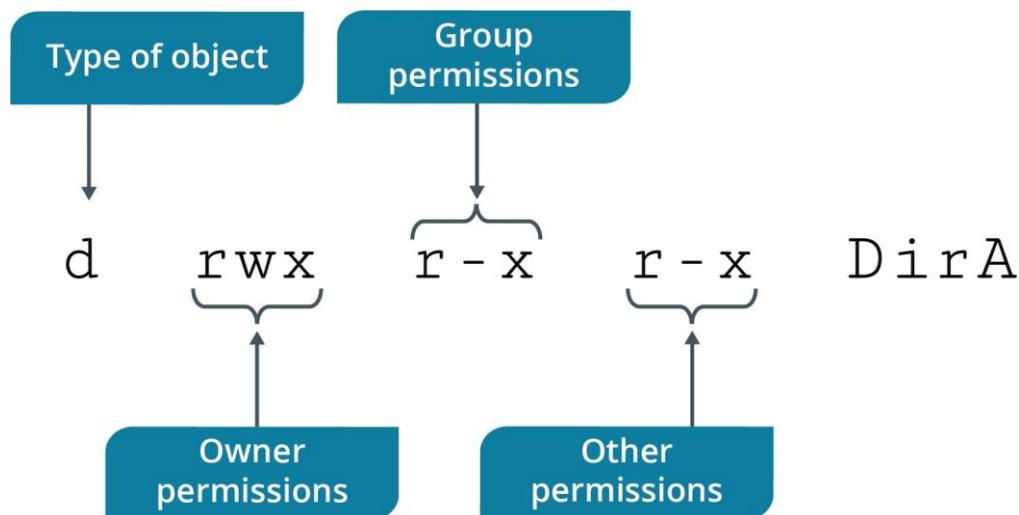
PERMISSION CONTEXTS

Permission attributes on files and folders are applied to one of several contexts, or the types of users and entities that you can give permission to. These contexts are:

- **Owner (u):** The owner of the file or directory, also simply referred to as the user.
- **Group (g):** The file or directory's group and all users belonging to that group.
- **Other (o):** All other users (neither owner nor group member).

PERMISSION STRING

The output of the `ls -l` command shows the **permission string** for a file or directory. The permission string contains 11 characters.



The permission string of a file. Note that special permissions and other information are not shown.

- The first character indicates the type of file; **d** for directory and hyphen (**-**) for file.
- Characters at the second, third, and fourth positions denote owner permissions.
- Characters at the fifth, sixth, and seventh positions denote group permissions.
- Characters at the eighth, ninth, and tenth positions denote other permissions.

- The final character indicates the access method for the file; period (.) for SELinux security context and plus (+) for any other combination of alternative access methods.

THE chmod COMMAND

The **chmod** command enables you to modify the permissions of a file or directory. Only the owner of the file or directory or the system administrator can change the permissions of the object.

SYNTAX

The syntax of the **chmod** command is **chmod [options] {mode} {file/directory name}**

chmod COMMAND OPTIONS

The **chmod** command supports different options to modify permissions. One or more of these options may be used at a time.

Option	Used To
-C	Report changes that are made in permissions.
-f	Hide most error messages.
-v	Display a diagnostic entry for every file processed.
-R	Modify permissions of files and directories recursively.

chmod SYMBOLIC MODE

The **chmod** command supports two modes: symbolic mode and absolute mode. Symbolic mode enables you to set permissions using three components, namely:

- Permission contexts: **u/g/o/a** (a applies the permissions to all three contexts).
- Permission operators: **+/-/=**
- Permission attributes: **r/w/x**

Permission operators determine whether a permission is to be granted or removed.

Operator	Description
+	Grants permissions.
-	Denies permissions.
=	Assigns permissions exactly as provided, rather than being additive or subtractive.

The screenshot shows a terminal window with the following commands:

```
root@server01 ~]# chmod g+r file1
[root@server01 ~]# chmod o-r file1
[root@server01 ~]# chmod go+rw file1
[root@server01 ~]#
```

Annotations explain the permissions being set:

- Gives group read permission to file**: Points to the first command `chmod g+r file1`.
- Removes read for others**: Points to the second command `chmod o-r file1`.
- Gives group and others read and write**: Points to the third command `chmod go+rw file1`.

Examples of setting permissions using symbolic mode.

SYNTA X

In symbolic mode, the syntax of the `chmod` command is:

```
chmod {access context}{operators}{permission attributes} {file/  
directory names}
```

As an example, to add read and write permissions to **myfile** for the owner and the group:

```
chmod u+rw,g+rw myfile
```

chmod ABSOLUTE MODE

The other `chmod` mode, absolute mode, uses **octal** (base-8) numbers to specify permissions. Each permission (r/w/x) has an associated number.

Octal Number	Attribute
4	Read
2	Write
1	Execute

By adding the octal numbers for the permissions you want to grant, you get the overall permission number to assign to a directory or file. For example, full permissions (read, write, and execute) are equivalent to $4 + 2 + 1$, or 7. Read and write permissions are equivalent to $4 + 2$, or 6. Complete permissions are expressed as a three-digit number, where each digit corresponds to the owner, the group, and others, respectively.

```
root@server01:~#
[root@server01 ~]# chmod 764 file1
[root@server01 ~]# chmod 700 file1
[root@server01 ~]# chmod 640 file1
[root@server01 ~]#
```

User = rwx
Group = rw
Others = r

User = rw
Group = r
Others = -

User = rwx
Group = -
Others = -

Examples of setting permissions using absolute mode.

SYNTAX

In absolute mode, the syntax of the `chmod` command is `chmod {number} {file/directory name}`

COMMON PERMISSIONS IN DIFFERENT MODES

The following table compares how the different `chmod` modes represent commonly assigned permissions.

Absolute Mode	Symbolic
755	u=rwx,g=rx,o=rx
700	u=rwx,g=,o=
644	u=rw,g=r,o=r
600	u=rw,g=,o=

THREE-DIGIT AND FOUR-DIGIT MODES

When written in octal, numeric format, file permissions typically have three digits, each digit corresponding to the user, group, and others permissions. However, file permissions may also be written with four digits, with the new, leading digit signifying any advanced permissions to be defined (or 0, for none). For example, the base permissions for non-executable files in Linux are `rW-rW-rW-`, or 666. This is equivalent to the octal format of 0666.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 3-1

Modifying File and Directory Permissions

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You're concerned about how to protect files and directories on a Linux server. You will interpret the existing permissions of a few files, and then configure permissions for the file owner, the group, and all others for files and directories.

1. View the existing permissions for files and directories.
 - a) Enter **ls -l** to see the permissions string for files and directories in your home directory.
 - b) Review the permissions assigned, identifying which bits are configured for the owner, the group, and all others.
 - c) Enter **ls -l /etc/ssh/sshd_config** to view the permissions for this configuration file.
 - d) Use the space below to write down the permissions for the owner, group, and others.

 - e) Enter **ls -l /var/log/cron** to view the permissions for this log file.
 - f) Use the space below to write down the permissions for the owner, group, and others.

2. Create a test directory and file you can configure the permissions for.
 - a) Enter **mkdir permissions-demo** to create a directory in your home directory.
 - b) Enter **cd permissions-demo** to move to that directory.
 - c) Enter **mkdir DirA** to create a permissions demonstration directory.
 - d) Enter **touch file1** to create a permissions demonstration file.
 - e) Enter **ls -l** to view the current permissions on both objects.

3. Configure permissions for the test directory and file using absolute mode.
 - a) Enter **chmod 755 DirA** to set permissions on the directory.
 - b) Enter **ls -l** to see how the permissions have changed on the directory.
 - c) Enter **chmod 660 file1** to set permissions on the file.
 - d) Enter **ls -l** to see how the permissions have changed on the file.
 - e) Enter **chmod 750 DirA** to set different permissions on the directory.
 - f) Enter **ls -l** to see how the permissions have changed on the directory.
 - g) Enter **chmod 744 file1**

- h) Enter **ls -l** and note the permissions changes.
4. Configure permissions for the test directory and file using symbolic mode.
- Enter **chmod o+r DirA** to set permissions on the directory.
 - Enter **ls -l** to see how the permissions have changed on the directory.
 - Enter **chmod go+rwx file1** to set different permissions on the file.
 - Enter **ls -l** to see how the permissions have changed on the file.
 - Enter **chmod go-rwx DirA**
 - Enter **ls -l** and note the permissions changes.
 - Enter **chmod go-w file1**
 - Enter **ls -l** and note the permissions changes.

The final permissions state of the directory should be **drwx-----**

The final permissions state of the file should be **-rwxr--r--**

DEFAULT PERMISSIONS

In Linux, default permissions are assigned to newly created files and directories based on user privileges. For files created by the root user, the default permissions are **644**, which means that the root user has read and write permissions, while group users and others will have only read permission. For directories created by the root user, the default permissions are **755**, which means that the root user has read, write, and execute permissions, while group users and others will have only read and execute permissions. In the case of users with limited access rights, Linux assigns permissions of **664** for newly created files and **775** for newly created directories.

These default permissions are determined by the user file creation mask, or umask. However, the default permissions may be altered by the root user.

THE umask COMMAND

The **umask** command alters the default permissions on newly created files and directories. Changing default permissions can be useful if you'd like to automatically control how new objects can be used, rather than changing these permissions manually on every new object.

With **umask**, you set default permissions using octal numeric format. However, instead of specifying which permissions to set, you specify which permissions to mask, or clear, from the default. For example, the default permissions for non-executable files in Linux are **666** (**rwx-rwx-rw-**). If you want the owner to retain these permissions, but you want the group and others to only have read access, you'll need to set the umask to **022**. Each number is explained as follows, in order:

- 0 means that the current owner permissions should not be masked at all, i.e., left as read and write.
- 2 means that the group permissions should be masked by 2, i.e., subtract 2 from the default (6) and you get 4. Group members now only have read access.
- 2 does the same thing as the previous number, but for other users.

You can use the **umask** command directly in the CLI to set the default permissions for that session, or you can set the default permissions for each user in their **.bashrc** file.



Note: Because it subtracts from the default (666), the `umask` command cannot force newly created files to set the execute bit.



System-wide umask:
set in `/etc/bashrc`
and `/etc/profile`

Default umask: set per-
user in `~/.bashrc`

umask set manually
with `umask` command

The order in which umask values are calculated.

SYNTA

X

The syntax of the `umask` command is `umask {number}`



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 3-2

Discussing File and Directory Permissions

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **Multiple users have complained about file access in a shared directory, but you've checked your daily backup reports and there are no corrupt files. Which command can you issue in the directory in question to investigate the problem further?**

2. **A group of system administrators were discussing file permissions and decided that setting a particular root-owned text file to read-only for everyone is a best practice. What do the permissions for this file look like?**

3. **A user cannot execute a script (`collect.sh`) she created and has sent you the contents of the script via email to inspect. After looking at the script, you determine the script is correctly written but permissions are the problem. What command can you issue to adjust the file's permissions as necessary?**

4. **A user changed the permissions of a script (`myscript.sh`) in a shared directory. The user is curious why everyone can execute the script if the user owns the script and everyone else only has read access. To make the script executable, what command did the user mistakenly issue?**

5. Your team lead is tired of receiving help desk tickets to restore deleted files from a directory that contains hundreds of files and subdirectories. She decides to have you fix the problem by making all of the files read-only. How do you change all the files to read-only without having to traverse each directory?
-

Activity 3-3

Modifying Default Permissions

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created a user account named **cmason**.

SCENARIO

One of the Develetech employees, Chris Mason, wants to create files and directories with non-default permissions so he can share them more easily with a co-worker. Since the requested change does not violate the Develetech security policy, it has been approved. You will implement the change for Chris.

1. View the current default permissions settings for users that create new files and directories.
 - a) Enter **umask**
 - b) Verify that the default mask is **0002**

For standard users, no advanced permissions are set by default (the first 0), owner and group permissions aren't masked, and other user permissions are masked by 2
2. Configure Chris Mason's **.bashrc** with a non-standard **umask** value.
 - a) Enter **sudo vim /home/cmason/.bashrc** to open the file in a text editor.
 - b) Press **Page Down** to move the cursor to the bottom of the file.
 - c) Press **i** to enter insert mode.
 - d) Add the following text on a new line:
umask 022
 - e) Press **Esc** to exit insert mode.
 - f) Enter **:WQ** to save and close the file.
3. Test the new default permissions.
 - a) Enter **SU - cmason** to switch credentials.
 - b) Enter **Pa22w0rd** when prompted.
 - c) Enter **umask** to view the current permissions default.

You should see the **0022** value configured above.
 - d) Enter **touch test-file** to create a new file.
 - e) Enter **ls -l** and verify that the permissions for **test-file** match the newly configured **umask** value.

The permissions should be **-rw-r--r--**
 - f) Enter **exit** to return to your student account.

Topic B

Modify File and Directory Ownership



EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Although you've restricted access to your data by assigning permissions, you may need to allow users to modify those permissions for certain files and directories. This is where the concept of ownership comes into play.

OWNERSHIP

As you've seen, the first permission context is the owner, or user. In Linux, **ownership** refers to the property by which a user can apply and modify the permissions of a file or directory. By default, the owner of a file or directory is the user who created that file or directory. Other than the superuser, no other user can change the permissions of an object that is owned by someone else.

While the most common application of ownership is the changing of read, write, and execute permission attributes, owners can also alter advanced and special permissions of the objects they own.

THE chown COMMAND

The **chown** command is used to change the owner, the group, or both for a file or directory. At times, you may wish for someone else to manage an object's permissions other than the user who created that object.

The following table describes how to use this command.

Command Syntax	Description
<code>chown {user name} {file/directory name}</code>	Changes the owner but not the group.
<code>chown {user name}:{group name} {file/directory name}</code>	Changes the owner and the group.
<code>chown {user name}:{file/directory name}</code>	Changes the owner and the group. The group will be changed to the specified user's login group.
<code>chown :{group name} {file/directory name}</code>	Changes the group but not the owner. This is the same as using the chgrp command.

RECURSIVELY CHANGING OWNERSHIP

You can combine the **chown** command with the **-R** option to recursively change ownership through a directory structure.

THE chgrp COMMAND

The `chgrp` command is used to change the group ownership of a file or directory. Changing the group ownership of an object ensures that the group permissions are applied to the specific group.

SYNTAX

The syntax of the `chgrp` command is `chgrp {group name} {file/ directory name}`



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 3-4

Discussing File and Directory Ownership

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **A user (Bob Smith—username: bsmith) calls you to request that you restore a group of files he accidentally deleted from his home directory. You copy the files for him but he later complains that he can no longer edit the files. What do you need to do so that he can edit his files?**
2. **Gina wants to share some marketing files with two other members of her team but doesn't want them to access those files in her home directory. She also wants the directory and its files to only be available to the Marketing group. What steps can you take as an administrator to accomplish this request?**
3. **What command is equivalent to issuing chown :mygrp file1?**
4. **As an administrator, you've created a shared directory and made the necessary permissions changes to allow a group of users access to that directory. You've also added users to the group. When a user (susan) who is a member of the group creates a file in the shared directory, what user and group permissions will the new file have?**

- 5. What must a user do in a shared directory to ensure that each group member has full read and write access to files they create?**
-

Activity 3-5

Modifying File and Directory

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You will create a Graphics department directory where department members can store content. You will investigate default ownership and group associations, and then create the **/Graphics** directory. You will configure ownership and group associations of the directory and files.

1. View the current ownership and group associations for files and directories.
 - a) Enter **ls -l /var/log/cron** to view ownership and group details about this log file.
The root user is the owner, and the root group is the group.
 - b) Enter **ls -l /etc/ssh/sshd_config** to view ownership and group details about this configuration file.
The root user is the owner, and the root group is the group.
 - c) Enter **sudo ls -l /home/cmason** to view ownership and group details about the contents of a user's home directory.
The **cmason** user is the owner, and the **cmason** group is the group.
2. Create a directory and populate the directory with files, then manage the ownership values.
 - a) Enter **sudo mkdir /Graphics** to create a directory at the root of the file system.
 - b) Enter **sudo touch /Graphics/file1** to create content.
 - c) Repeat this command with **file2** and **file3** to create additional empty files inside the directory.
 - d) Enter **ls -l /Graphics** to view the ownership information.
The owner is the creator; in this case that is the root account, due to the use of the **sudo** command.
 - e) Enter **sudo chmod -R 774 /Graphics** to set permissions on the **/Graphics** directory and its contents.
 - f) Enter **sudo ls -l /Graphics** to view the new permissions.

```
[student01@localhost permissions-demo]$ sudo ls -l /Graphics
total 0
-rwxrwxr--. 1 root root 0 Dec 14 13:06 file1
-rwxrwxr--. 1 root root 0 Dec 14 13:06 file2
-rwxrwxr--. 1 root root 0 Dec 14 13:06 file3
```

3. Change the owner and group values of the **/Graphics** directory and its contents.
 - a) Enter **sudo chown -R :GraphicsDept /Graphics** to set the group association as **GraphicsDept**
 - b) Enter **sudo ls -ld /Graphics** to view the changes.
 - c) Enter **sudo chown rstanley /Graphics/file2** to change the ownership of **file2** to Rose Stanley.
 - d) Enter **sudo ls -l /Graphics** to confirm **rstanley** is now the owner of **file2**
-

Topic C

Configure Special Permissions and Attributes



EXAM OBJECTIVES COVERED

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

The standard read, write, and execute permissions are good enough in most circumstances. However, there are additional permissions and attributes that you can use to restrict access in a more specialized way.

SPECIAL PERMISSIONS

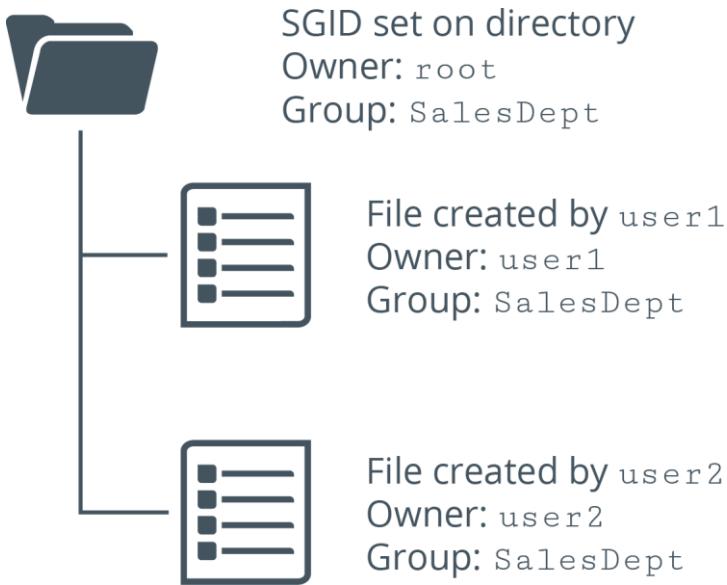
Special permissions are used when normal permissions become inadequate. With special permissions, less privileged users are allowed to execute a file by assuming the privileges of the file's owner or group. This enables the user to temporarily take on these higher level privileges in a limited context.

SUID AND SGID PERMISSIONS

In Linux, two main special permissions are set user ID (SUID) and set group ID (SGID). **SUID**, or setuid, is the permission that allows a user to have similar permissions as the owner of the file. Certain executable files and commands, like `passwd`, require access to additional resources that the user may not have explicit permissions to. Instead of granting those permissions individually, the `passwd` command is configured so that any user will execute it as root (the owner), giving them permission to the additional resources.

SGID, or setgid, is the permission that allows a user to have similar permissions as the group owner of the file. In addition to files, SGID can also be set on directories. Any subdirectories created in this directory will automatically **inherit** the SGID permission. Likewise, all new files and subdirectories created in this directory will inherit the directory's group ID, rather than the group ID of the user who created the object. This inheritance is useful because users in a shared environment don't need to change their group when they create objects in the directory. Note that the SGID permission is not applied to existing objects in the directory, nor is it applied to objects that are moved from other locations into the directory.

SUID and SGID are both set using the `chmod` command, and you can do so using either symbolic mode or absolute mode. When using `ls -al` to see permissions, the execute permission for the owner will appear as `S` for the SUID, and the execute permission for the group will appear as `S` for the SGID.



An example of setting SGID on a directory.

SYNTAX

The following is the syntax for setting the SUID on a file, using symbolic and absolute mode, respectively:

```
chmod u+s {file names}
chmod 4### {file names}
```

Note the last three bits in absolute mode are whatever standard permissions you choose.

The following is the syntax for setting the SGID on a directory, using symbolic and absolute mode, respectively:

```
chmod g+s {directory names}
chmod 2### {directory names}
```

Removing the SUID and SGID is as simple as using the **-** (minus) operator in symbolic mode, or setting the first permission bit to 0 in absolute mode.

STICKY BIT

A **sticky bit** is a special permission bit that provides protection for files in a directory. It ensures that only the owner of a file or directory (or root) can delete the file or directory. Without the sticky bit, any user with write and execute permissions on an object can delete that object. The sticky bit ensures that these users do not have delete privileges, but still have the rest of the privileges that come with writing and executing files and directories.

Like SUID/SGID, you set a sticky bit using the **chmod** command. Using **ls -al** you can see the sticky bit in the execute position for other users (the last position) as the letter **t**, or the capitalized letter **T** if the execute permission is not set for others.

SYNTAX

The syntax for setting the sticky bit is as follows, using symbolic mode and absolute mode, respectively:

```
chmod +t {directory names}
chmod 1### {directory names}
```

As with SUID/SGID, use – or 0 to clear the sticky bit.

STICKY BIT ON FILES

In older versions of the kernel, a sticky bit could force a program or file to remain in memory so that it wouldn't need to be reloaded when it was invoked again. A sticky bit on a file indicated to the operating system that the file would be executed frequently. Modern versions of the Linux kernel ignore the sticky bit on files; if you want to protect specific files, you need to apply the sticky bit on the directory that contains them.

FILE ATTRIBUTES

Files can have one or more attributes set on them that define how the system interacts with those files. These attributes go beyond typical permissions and enable you to more granularly customize what the system is and is not allowed to do with a file.

There are many such attributes. Some examples include:

- Only allow the file to be open for writing in append mode; i.e., don't allow the file to be overwritten.
- Set the file to be automatically compressed.
- Save the file if it is deleted, providing an opportunity for it to be recovered.
- Make the file immutable.

THE IMMUTABLE FLAG

The **immutable flag** is an attribute of a file or directory that prevents it from being modified, even by the root user. In other words, no one can delete, rename, or write to an immutable file. Setting the immutable flag is useful for files with a high degree of sensitivity and importance, and which are also not likely to change any time soon. A careless user or an errant process will be unable to delete the immutable file.

The immutable flag is not set on all files. A single directory can have a mix of mutable and immutable files and subdirectories. Also, an immutable subdirectory can have mutable files.

When viewing file attributes, the lowercase **i** character indicates that the immutable flag is set.

THE **Isattr** COMMAND

The **Isattr** command is used to list the attributes of a file or directory.

The following table describes some of the options of the **Isattr** command.

Option	Used To
-R	Recursively list the attributes of directories and their contents.
-a	List all files in directories.
-d	List directories like files, instead of listing their contents.
-v	List the version number of the file.

```
root@server01:~# lsattr file1
-----i----- file1
```

Immutable attribute set

Listing the attributes of a file.

SYNTAX

The syntax of the **lsattr** command is **lsattr [options] {file/directory names}**

THE chattr COMMAND

The **chattr** command is used to change the attributes of a file or directory. The following table describes some of the options of the **chattr** command.

Option	Used To
-R	Recursively change the attributes of directories and their contents.
-v {version}	Set the version number of a file.
+i	Mark the file as read-only, or immutable. Requires superuser privileges.
-i	Remove the read-only, or immutable, attribute of the file. Requires superuser privileges.



Note: Only the root user can set or remove the immutable flag.

SYNTAX

The syntax of the **chattr** command is **chattr [-R] [-v {version}] [+{-attributes}] {file/directory names}**

ACCESS CONTROL LISTS

An **access control list (ACL)** is a list of permissions attached to an object. ACLs can be used for situations where the traditional file permission concept does not suffice. ACLs enable you to assign permissions to individual users or groups even if these do not correspond to the object's owner or group.

For example, members of two department groups may need different levels of access to the same resource. Group 1 might need **r/w/x** to a directory, whereas Group 2

only needs **r/x** access. By using ACLs, you are able to grant different levels of access to different users, groups, and even processes. ACLs enable a more granular level of control.

THE getfacl COMMAND

The **getfacl** command is used to retrieve the ACLs of files and directories.

The basic output format of the **getfacl** command shows metadata about the object including its owner, its group, any SUID/Sgid/sticky bit flags set, the standard permissions associated with the object, and the individual permission entries for users and groups.

```
root@server01 ~]# getfacl file1
# file: file1
# owner: root
# group: root
# flags: --t
user::rw-
user:manderson:rw- ← Individual user permissions
group::r--
mask::rw-
other::---
```

An ACL that sets permissions for a specific user.



Note: Directories can also have default ACL entries that pertain to any new files or subdirectories created within them.

THE setfacl COMMAND

The **setfacl** command is used to change the permissions associated with the ACL of a file or directory.

The **setfacl** command has several options, some of the most common of which are described in the following table.

Option	Used To
-R	Recursively set ACL options for directories and their contents.
-S	Set the ACL of an object, replacing any existing ACL.
-M	Modify the existing ACL of an object.
-X	Remove entries from an existing ACL.
-B	Remove all ACL entries (not including the standard permissions).

SYNTAX

The syntax of the **setfacl** command is **setfacl [-bR] [-mx {acl_spec}] {file/directory names}**

ACL SPECIFICATION

The ACL specification can be formatted in one of several ways:

- When working with users, the format is u:{user name}:{permissions}
- When working with groups, the format is g:{group name}:{permissions}

EXAMPLE

The following is an example of modifying the ACL on a directory where the user **http** is given read access:

```
setfacl -m u:http:r-- /home/directory
```



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 3-6

Discussing Special Permissions and Attributes

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **The Marketing manager contacts you stating that the shared directory you set up for the Marketing group works, but not exactly like they'd planned. When one of the group members creates a file in the directory, the file takes on the user's user and group permissions. For example, when Linda creates a new file, the permissions are `-rw-rw-r-- linda linda`. The manager wants the files to all retain the mkt group permission, if possible, rather than having the users change the group themselves. What action can you take to fulfill this request?**

2. **Gina, a member of the Marketing group, has decided that she wants her files protected so that only she can delete them, although other Marketing group members need to be able to work on and edit the files. What can she do to fix the problem of others deleting her files?**

3. **Ruth has searched for a solution to her problem: A few of her training documents keep getting changed or removed by system administrators removing files that haven't been accessed in excess of 180 days. She has found that a file can be made immutable, but she cannot make her own files immutable and needs your assistance. How can you make the files `/home/ruth/training1_doc.txt` and `/home/ruth/training2_doc.txt` immutable?**

- 4. A user, John, opened a ticket complaining that he has files in his home directory that he cannot remove, although they are his files and he is the user and group owner. He requests that you remove the files /home/john/billing1.txt, billing2.txt, and summary.txt. However, when you attempt to remove the files, you receive an error that you cannot remove the files as the root user. What is a possible resolution for John?**

 - 5. You created a shared directory for the Marketing planners, Linda and Mark, named /opt/MPlans. Their group, mplan, has exclusive access to this directory. No other user can access the directory or its contents. Linda decides that Gina needs read-only access to a single file, history.txt, inside the /opt/MPlans directory. Is this kind of restrictive access possible? If so, how can you grant it to Gina?**

Activity 3-7

Configuring SGID Permissions and Sticky Bits

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created a `/Graphics` directory as well as the `GraphicsDept` group that includes several users.

SCENARIO

Some users have noted that the group associations for `/Graphics` are not applied to files created in the directory. One user also complained that another user accidentally deleted one of her files. You are asked to correct these concerns.

1. Use SGID to automatically set group associations for newly created files in the `/Graphics` directory.
 - a) Enter `ls -ld /Graphics` to see the default permissions on the `/Graphics` directory.
 - b) Enter `sudo chmod g+s /Graphics` to set the SGID on `/Graphics` so that newly created files will get the group association.
 - c) Enter `ls -ld /Graphics` to view the new permissions.

```
drwxrwsr--. 2 root GraphicsDept 45 Dec 14 13:06 /Graphics
```

 - d) Enter `SU - rstanley` and enter `Pa22w0rd` to switch to Rose Stanley's credentials.
 - e) Enter `Cd /Graphics` and then `touch file4` to create a file.
 - f) Enter `ls -l` and confirm `rstanley` is the owner and the group is `GraphicsDept` for `file4`.
 - g) Enter `exit` to return to the `student##` login.
2. Use the sticky bit to better protect files from deletion by anyone but their owner.
 - a) Enter `sudo chmod +t /Graphics` to configure the sticky bit on the `/Graphics` directory.
 - b) Enter `SU - jrobinson` and enter `Pa22w0rd` to switch to Jerry Robinson's credentials.
 - c) Enter `cd /Graphics` to move to the `/Graphics` directory.
 - d) Enter `rm file4` to attempt to delete the file owned by `rstanley`.

Note that you receive an "Operation not permitted" response. If this were a permissions issue, you would receive an "access denied" response instead. Even though `jrobinson` is a member of the `GraphicsDept` group, and that

group has the permissions to delete a file in this directory, the sticky bit is preventing file deletion from a non-owner.

- e) Enter **exit** to return to the **Student##** login.
-

Activity 3-8

Setting the Immutable Flag on a File

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You have written a README text file to be stored in the **/Graphics** directory to help guide users on the proper use of the content. You want to ensure that no one, not even the root user, can accidentally delete the file. You will use the immutable attribute to accomplish this task.

1. Configure the immutable flag on the **/Graphics/README** file.
 - a) Enter **sudo touch /Graphics/README** to create a document in the **/Graphics** directory.
 - b) Enter **sudo ls -l /Graphics** to view the current permissions settings for the **README** file.
 - c) Verify the **README** file is owned by root, and that the owner would normally be able to delete the file.

```
-rw-r--r--. 1 root      GraphicsDept 0 Dec 14 13:15 README
```

-
- d) Enter **sudo chattr +i /Graphics/README** to set the immutable attribute on the **README** file.
- e) Enter **sudo ls -l /Graphics** to view the current permissions and verify that they haven't changed.
- f) Enter **sudo lsattr /Graphics/README** to confirm the immutable attribute is set.

```
-----i----- /Graphics/README
```

2. Verify that the flag works by attempting to delete the file.
 - a) Enter **sudo rm /Graphics/README** to attempt to delete the **README** file from the **/Graphics** directory.
 - b) Verify that this fails.
- This is due to the immutable attribute. Note the "Operation not permitted" response rather than the "access denied" response that indicates a permissions issue.

Activity 3-9

Configuring ACLs

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

The Graphics department has requested that the Marketing department be given read-only access to the **/Graphics** directory. With standard permissions, only one group association can exist. You will use access control lists (ACLs) to ensure that both the Graphics and Marketing departments have access.

Set an ACL for the Marketing department.

- a) Enter **sudo getfacl /Graphics** to view the current ACL on the directory.
 - b) Enter **sudo setfacl -R -m g:MarketingDept:r /Graphics** to grant read-only permissions to the **MarketingDept** to the **/Graphics** directory and its contents.
- Note:** You can ignore the "Operation not permitted" warning about the **README** file; the ACL settings will still apply to all other objects.
- c) Enter **sudo getfacl /Graphics** to view the new level of access for the **MarketingDept**.

```
# file: Graphics
# owner: root
# group: GraphicsDept
# flags: -st
user::rwx
group::rwx
group:MarketingDept:r--
mask::rwx
other::r--
```

Topic D

Troubleshoot Permissions Issues



EXAM OBJECTIVES COVERED

4.3 Given a scenario, analyze and troubleshoot user issues.

In any system, problems are inevitable. As a Linux professional, one of the most important skills you'll need is the ability to troubleshoot these problems. So, in this topic, you'll follow a general model for troubleshooting any type of operating system problem. You'll then put these troubleshooting skills to use by diagnosing and solving issues related to permissions.

TROUBLESHOOTING

Troubleshooting is the recognition, diagnosis, and resolution of problems.

Troubleshooting begins with the identification of a problem, and it does not end until services have been restored and the problem no longer adversely affects users.

Troubleshooting can take many forms, but all approaches have the same goal: to solve a problem efficiently with a minimal interruption of service.

TROUBLESHOOTING MODELS

A troubleshooting strategy is a plan of action for identifying the causes and resolving the effects of a system-related issue. You can implement a troubleshooting strategy through a step-by-step approach, or a **troubleshooting model**. There are many such models, and they can vary in their approach, but all models attempt to enable the troubleshooter to move in a methodical and repeatable pattern during the troubleshooting process.

One example of a troubleshooting model divides the process into the following steps:

1. Identify the problem. This stage includes:
 - Gathering information.
 - Duplicating the problem, if possible.
 - Questioning users to gain experiential information.
 - Identifying the symptoms.
 - Determining if anything has changed.
 - Approaching multiple problems individually.
2. Establish a theory of probable cause. This stage includes:
 - Questioning the obvious.
 - Considering multiple approaches.
 - Looking for not just a cause, but the root cause.
3. Test the theory to determine the cause.
 - When the theory is confirmed, determine the next steps to resolve the problem.
 - If the theory is not confirmed, establish a new theory or escalate the issue.
4. Establish a plan of action to resolve the problem, while identifying the potential effects of your plan.

5. Implement the solution, or escalate the issue.
6. Verify full system functionality and, if applicable, implement preventative measures.
7. Document your findings, actions, and the outcomes.

PERMISSIONS TROUBLESHOOTING

As a general guideline, whenever you are denied permissions when you expect to have them, or vice versa, you should always verify the permissions of the relevant object by using the `ls -al` command. That will often save you from a lot of guesswork, as many issues simply come down to the wrong permission being applied.

The following table lists some common symptoms that relate to permissions issues, as well as some potential causes and solutions. Note that these are examples, and that some of these symptoms can have multiple causes and solutions.

Symptom	Cause and Solution
The owner of a text file is denied permission to view the contents of the text file.	<p>Cause: Despite being the owner, the user is not explicitly granted read access.</p> <p>Solution: Use <code>chmod</code> to grant read privileges to the owner context.</p>
A user is denied permission to remove a directory, despite having write permission.	<p>Cause: The user needs both write <i>and</i> execute permission to modify a directory.</p> <p>Solution: Use <code>chmod</code> to add execute permission to the directory for the appropriate context.</p>
A user is denied permission to enter into a directory, despite having read permission.	<p>Cause: The user needs execute permission to change to a directory.</p> <p>Solution: Use <code>chmod</code> to add execute permission to the directory for the appropriate context.</p>
A user is denied permission to remove a file, despite having full permissions on that file.	<p>Cause: The user must have write permission on the containing directory.</p> <p>Solution: Use <code>chmod</code> to add write permission to the directory for the proper context.</p>
A user is denied permission to create files in a directory they have write and execute permission to.	<p>Cause: The immutable flag is set on the directory.</p> <p>Solution: As root, remove the immutable flag from the directory using the <code>chattr</code> command.</p>
The root user is denied permission to modify a file.	<p>Cause: The immutable flag is set on the file.</p> <p>Solution: As root, remove the immutable flag from the file using the <code>chattr</code> command.</p>

Symptom	Cause and Solution
All users have the ability to list the contents of a directory, when only the owner, group members, and a specific service account should.	<p>Cause: The read permission is set on the directory for the others context.</p> <p>Solution: Remove read permission from the others context. Add the service account to the directory's ACL using the setfacl command, granting the account read access.</p>
A user is denied permission to execute a script that they themselves created.	<p>Cause: The execute permission is not automatically set for new files.</p> <p>Solution: Use chmod to add execute permission to the script for the file owner.</p>
A user is denied permission to change the contents of a script, even though they have the ability to execute that script.	<p>Cause: Like with any other file, the user needs write permission in order to change a script's contents.</p> <p>Solution: Use chmod to add write permission to the script for the appropriate context.</p>
A user is denied permission to execute a script, despite having execute permission.	<p>Cause: The user also needs read permission in order to execute a script.</p> <p>Solution: Use chmod to add read permission to the script for the appropriate context.</p>
All users are able to delete a file, but they should only be able to write to it.	<p>Cause: By default, the write and execute permissions on directories enable users to delete the objects therein.</p> <p>Solution: Add the sticky bit permission to the container directory so that only the owner or root can delete the file.</p>

OWNERSHIP TROUBLESHOOTING

Just like with permissions, you should use **ls -al** to verify the user and group ownership of a file or directory whenever you are experiencing ownership-related issues. Beyond that, the following table lists some potential issues you may come across, as well as suggested causes and solutions.

Symptom	Cause and Solution
A user is unable to access a file, despite the owner context having full permissions.	<p>Cause: The user is not the owner of the file.</p> <p>Solution: Use the chown command to make the user the owner of the file.</p>
A user is unable to delete a file, despite the containing directory granting full permissions to the group.	<p>Cause: The directory's owning group is not the same as the user's group.</p> <p>Solution: Use the chgrp command to make the directory's owning group the same as the user's.</p>

Symptom	Cause and Solution
Several users are able to modify a file despite the others context only having read permission.	Cause: The file's owning group is set to the same group that these users are members of. Solution: Use <code>Chgrp</code> to change the file's owning group to some other group.
When a user creates files in a shared directory, the files take on the user's group ID, when they should take on the directory's group ID.	Cause: By default, files created by a user take on that user's group ID. Solution: Use <code>chmod</code> to set the SGID permission on the containing directory so that all new files inherit the directory's group ID.
When a user creates files in a shared directory, the files take on the directory's group ID, when they should take on the user's group ID.	Cause: The SGID permission is set on the shared directory so that new files inherit the directory's group ID. Solution: Use <code>chmod</code> to remove the SGID permission on the containing directory, disabling inheritance of the directory's group ID.

GROUP MEMBERSHIP TROUBLESHOOTING

Some issues arise because, despite having configured permissions and ownership correctly, the user may not be placed in the correct group. Use the `groups {user name}` command to discover what groups a user is a member of. A related issue is that, when a user creates files, the default owning group is not what is expected. Make sure the expected group is the user's primary group, rather than a secondary group. In either case, use the `Usermod` command to change group membership when applicable.

It may also be beneficial to list all of the members of a group so you identify which accounts to add or remove as necessary. However, there is not necessarily one simple command that is universal to Linux distributions that can accomplish this. You can search the `/etc/group` file for the desired group, but this only displays groups in the standard database (i.e., not other authentication methods), and it doesn't show users whose primary group is the group you're searching for. The `lid` and `libuser-lid` commands are pre-installed on some distributions and can retrieve all members of a group, including members whose primary group is the group being searched for. The `getent` command, available on some distributions, enables you to retrieve group members of non-standard authentication methods.

GUIDELINES FOR TROUBLESHOOTING PERMISSIONS ISSUES

Use the following guidelines when troubleshooting permissions issues.

TROUBLESHOOT PERMISSIONS ISSUES

When troubleshooting permissions issues:

- Follow an overall troubleshooting strategy for any kind of troubleshooting task.

- Follow a step-by-step troubleshooting model that can produce repeatable and demonstrable results.
- Start by verifying an object's permissions and ownership using the **ls -al** command.
- Ensure users have the permissions to work with a file if they are being denied access.
- Ensure users do not have permissions that enable them to access files beyond what they should have.
- Ensure objects don't have the immutable flag set if you expect to modify them.
- Set the SUID permission on an executable if you need to run it with root permissions for it to function properly.
- Set the sticky bit when you only want the owner and root to be able to delete an object.
- Ensure objects have the proper owner and owning group set.
- Set the SGID permission on a directory when you want new files to take on that directory's group ownership.
- Use the **groups {user name}** command to verify the groups a user is a member of.
- Modify group membership when necessary to ensure access or a lack thereof for specific users.
- Acquire and use tools like **lid** and **getent** to view members of a specific group.

Activity 3-10

Troubleshooting Permissions Issues

SCENARIO

You are working with the Develetech service desk to complete some Linux service requests. You will troubleshoot permissions tickets. You must identify a probable cause for the tickets, and suggest a resolution.

1. **Rose Stanley opened a ticket indicating that she is denied the ability to delete a directory. She confirmed with the service desk that she has write permission to the directory. What suggestion would you make to address the issue?**

2. **Jerry Robinson received an access denied message on a file. He told the service desk that the user permissions indicate the owner has read access. What suggestion would you make to address the issue?**

3. **Rose Stanley received an access denied message on a file. She used the `ls -l` command to discover what group had been granted access to the file. She believed she should be able to access the file. What suggestion would you make to address the issue?**

4. **Jerry Robinson cannot use the `cd` command to navigate into a particular directory. He told the service desk that he used `ls -l` to see what group is associated with the file. He also confirmed that the group is listed as having read access to the directory. He then used the `id` command to confirm he is a member of that group. What suggestion would you make to address this issue?**

5. **A group with a shared directory set up by another administrator is having several problems with permissions in that directory. You've been asked to investigate. Which commands can you use to check existing permissions to help assess the current setup?**

6. **A user complains that a script that she created and has full access to (rwxrwx---) is not executing for her. What is the problem with this script?**

 7. **Bob opens a ticket stating that he receives a permission denied error when trying to cd into /opt/Finance. Bob is a member of the FinanceDept group. What is the problem with Bob's access?**

 8. **The application development team created a new web app and requested that you set up a web server-accessible directory and a service account for the application. After setting up everything for the team, they opened a ticket claiming that the application cannot write logs to the log directory. What is the issue, and how can you correct it?**

 9. **The Graphics department requests that everyone in the company have access to company logo art to insert into emails, letterheads, and other documents. They do not want anyone outside of their group, however, to have any other access. A few days after requesting the change, other users still cannot access the files. You check permissions on the shared directory to find that the permissions are as follows: drwxrwx--- graphics GraphicsDept 4096 Dec 1 09:42 logos —What command can you issue to fix the problem?**
-

Summary

In this lesson, you modified permissions and ownership of files and directories in a Linux system. Now, you will be able to more efficiently control access to data on your Linux systems.

In what situations might you want to change the default permissions on your Linux systems?

Do you prefer to set permissions using symbolic mode or absolute mode, and why?



Practice Question: Additional practice questions are available on the course website.

Lesson 4

Managing Storage

LESSON TIME: 3 HOURS, 30 MINUTES

LESSON INTRODUCTION

Aside from managing user access to your Linux® systems, one of the most foundational tasks is the management of data storage. There are many ways to divide, format, and otherwise organize how your data is stored, and Linux gives you many tools for doing so. Using these tools, you will ultimately make it easier for authorized users to work with data.

LESSON OBJECTIVES

In this lesson, you will:

- Create drive partitions.
- Manage volumes using the Logical Volume Manager (LVM).
- Mount Linux file systems.
- Manage Linux file systems.
- Navigate the directory structure defined in the Filesystem Hierarchy Standard (FHS).
- Troubleshoot storage issues.

Topic A

Create Partitions



EXAM OBJECTIVES COVERED

- 1.4 Given a scenario, manage storage in a Linux environment.
- 2.3 Given a scenario, create, modify, and redirect files.
- 4.1 Given a scenario, analyze system properties and remediate accordingly.

The first task in managing storage is to partition a storage device and format the partition with a file system. This will make the section of the device available for the reading and writing of data.

STORAGE DEVICES

A **storage device** is a physical component that can record data and hold it persistently. There are many types of storage devices that are supported by the Linux operating system. Common types include:

- **Hard disk drive (HDD)**—electromechanical devices that use magnetic storage technology to store data, usually in large amounts.
- **Solid-state drive (SSD)**—storage devices that use non-mechanical solid-state technology to store data, usually in large amounts. They tend to support much quicker access times than HDDs.
- **USB thumb drive**—portable storage devices that use flash memory technology to store data, usually in small amounts compared to HDDs and SSDs. Their small size makes them easy to move around.
- **External storage drive**—portable storage drives that can use one of several technology types. They usually connect to a computer through a peripheral interface like USB, rather than being an internal component.

Although the previous storage devices are most likely what you'll be working with, you might also be responsible for working with legacy technology like floppy disk drives (FDD), tape drives, etc.



Note: The terms "disk" and "hard drive" are often used to refer to all storage devices generally. Even some Linux tools do this. However, not all storage devices use hard disk technology, so this course uses terms like "storage device," "storage drive," and "storage media" to refer to the general concept.

BLOCK VS. CHARACTER DEVICES

Linux refers to devices as either block or character devices. Block devices are storage devices (like those listed previously) that can be read from and written to in blocks of data. Character devices are devices like keyboards, mice, serial ports, etc., that can be read from and written to in streams of data.

FILE SYSTEMS

A **file system** is a data structure that is used by an operating system to store, retrieve, organize, and manage files and directories on storage devices. A file system maintains information such as the date of creation and modification of individual files, the size of files on the storage device, the type of files, and permissions associated with files. It also provides a structured form for data storage.

A file system by itself does not interpret the data contained in files because this task is handled by specific applications. File systems vary depending on several parameters, such as the purpose of the file systems, the information they store about individual files, the way they store data, and the data security mechanisms they implement.

TYPES OF FILE SYSTEMS

Linux supports many file system types. The most common are described in the following table.

File System Type	Description
FAT	File Allocation Table (FAT) is an older file system that is compatible with many different operating systems, including all versions of Unix, Windows, and macOS. It does not provide the same capabilities as more modern file systems and is typically used for compatibility reasons. Improved versions include FAT32 and exFAT.
ext2	This used to be the native Linux file system of some older releases. It is still supported in current releases of Linux.
ext3	This is an improved version of ext2. In case of an abrupt system shutdown, ext3 is much faster in recovering data and better ensures data integrity. You can easily upgrade your file system from ext2 to ext3.
ext4	This is one of two default file system for Linux distributions. It is backwards-compatible with the ext2 and ext3 file systems. Among ext4's improvements over ext3 are journaling, support of volumes of up to one exbibyte (EiB), and files up to 16 tebibyte (TiB) in size. This is the default file system for Ubuntu® installations.
XFS	This is a 64-bit, high-performance journaling file system that provides fast recovery and can handle large files efficiently. XFS is the default file system for CentOS®/RHEL 7 installations.

NTFS

The **New Technology File System (NTFS)** is a proprietary file system created by Microsoft® as the primary file system for Windows®. NTFS provides many enhanced features over FAT, including file- and folder-level security, file encryption, drive compression, and scalability to very large drives and files.

Linux does not support NTFS by default; however, a utility called NTFS-3G can enable support for NTFS on Linux systems.

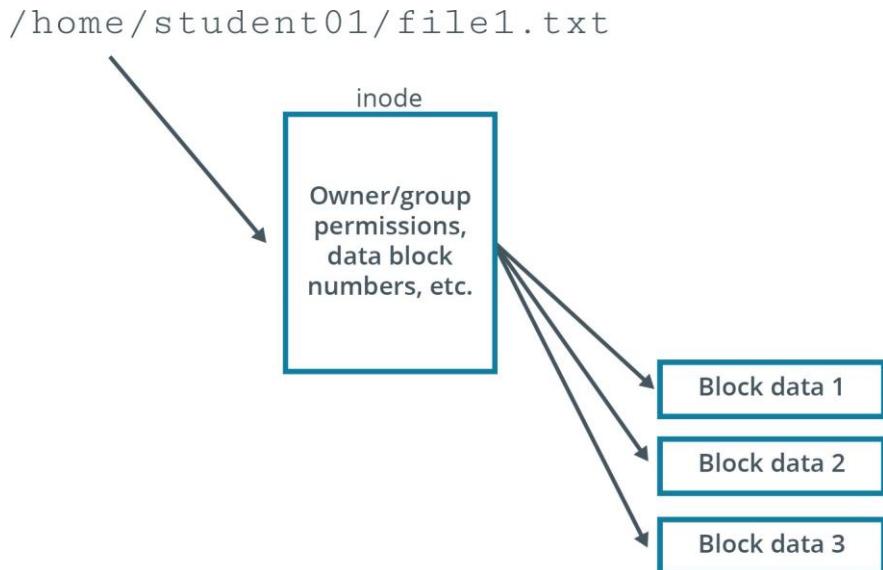
NETWORK FILE SYSTEMS

While the previous table lists general-purpose file systems, some file systems function as network protocols that enable the sharing of data over a network. Common types include the following.

Network File	Description
Server Message Block (SMB)	The SMB protocol provides users shared access to files and other resources across a local area network (LAN). SMB clients make requests for resources to SMB servers, which respond and provide the appropriate level of access. This protocol is primarily used with Windows computers. However, SMB-compatible software called Samba helps interface Linux and Windows hosts running network shares.
Common Internet File System (CIFS)	CIFS is a specific implementation of SMB that is rarely in use. Microsoft designed it as a successor to SMB version 1, but SMB versions 2 and 3 superseded it. However, Linux still uses the CIFS name in some of its tools, though these tools support newer versions of SMB.
Network File System (NFS)	NFS offers similar functionality to SMB, but the protocols are not compatible. NFS is preferred in situations where Linux clients access Linux servers. In environments that are a mix of Windows and Linux, the SMB protocol is the better choice.

INODES

An **index node (inode)** is an object that stores metadata about a file or directory on a file system. This metadata can include time-based values like when a file was created and last modified; permission and ownership information; the block locations of a file's data on a storage device; and other miscellaneous information.



A specific file with its associated inode.

Each inode on a file system is identified by a unique integer called an inode number. Whenever the system or an application tries to access a file, it searches for the appropriate inode number in a data structure called an inode table. The inode table maps an inode number to its corresponding file or directory name.

Some file systems set a maximum number of inodes when that file system is created, usually by considering the overall size of the file system. The total number of files and directories cannot exceed this inode maximum. However, some file system types, like XFS, use a dynamic inode allocation system that scales as a percentage of the file system's capacity. In other words, these file systems do not set a strict inode limit.



Note: You can use the `ls -i` command to list the inode numbers for files and directories.

JOURNALING

Journaling is a method by which a file system records changes that have not yet been made to the file system itself in an object called a journal. This enables the file system to quickly recover after an unexpected interruption, such as a system crash, because the system can reference pending changes in the journal to resume where it had left off.

The journaling process involves the following phases:

1. The journal describes all the changes that must be made to the drive.
2. A background process makes each change as and when it is entered in the journal.
3. If the system shuts down, pending changes are performed when it is rebooted.
4. Incomplete entries in the journal are discarded.

VIRTUAL FILE SYSTEM

A real file system refers to a discrete file system that the Linux kernel can normally work with directly. The problem is, Linux supports many different file system types that aren't necessarily compatible. The **virtual file system (VFS)** was created as a common software interface that sits between the kernel and real file systems. In effect, the VFS translates a real file system's details to the kernel so that the file system appears identical to any other file system.

With VFS, you can mount multiple different types of file systems on the same Linux installation, and they will appear uniform to the user and to all other applications. Therefore, the user and these applications can work with the file system without actually knowing its underlying structure. This greatly increases interoperability between the system and running software.

EXAMPLES

Examples of real file systems on a Linux system can include `xfs`, `ext4`, and several other types. Examples of virtual file systems can include `proc`, which contains system information during runtime; `devtmpfs`, which contains device nodes loaded by the kernel during system initialization; `debugfs`, which contains information useful in debugging the Linux kernel; and many more.

FILE SYSTEM LABELS

File system labels are assigned to file systems for easy identification. The labels may be up to 16 characters long and can be displayed or changed using the `e2label` command for ext# file systems and the `xfs_admin` command for XFS file systems.

SYNTAX

The syntax for setting ext# file system labels is `e2label /dev/{device name}{partition number} {label name}`

The syntax for setting XFS file system labels is `xfs_admin -L {label name} /dev/{device name}{partition number}`

PARTITIONS

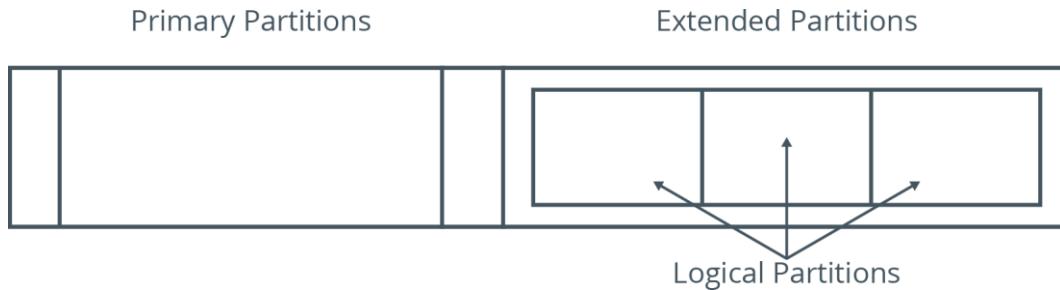
A **partition** is a section of the storage drive that logically acts as a separate drive. Partitions enable you to convert a large drive to smaller manageable chunks, leading to better organization of information. A partition must be formatted and assigned a file system before data can be stored on it.

Partitions are identified using a partition table, which is stored in one or more areas of the drive. The size of each partition can vary but cannot exceed the total free space of the storage drive.

Most operating systems, including Linux, use drive partitions. Data of different types can be stored in separate locations on the drive, such as separating system files from user-accessible files.

TYPES OF PARTITIONS

There are three types of partitions: primary, extended, and logical. The functionality of the storage drive depends on the types of partitions on it.



The layout of partition types on a block storage device.

Each partition has a set of specific features. The three types of partitions are described in the table.

Partition	Description
Primary	A partition that can contain one file system or logical drive and is sometimes referred to as a volume. The swap file system and the boot partition are normally created in a primary partition.
Extended	An extended partition can contain several file systems, which are referred to as logical drives. There can be only one extended partition, which can be further subdivided. This partition type does not contain any data and has a separate partition table.
Logical	A part of a physical drive that has been partitioned and allocated as an independent unit and functions as a separate drive. A logical partition is created within an extended partition, and is therefore a subset of an extended partition. There is no restriction on the number of logical partitions, but it is advisable to limit it to 12 logical partitions per drive.

SWAP SPACE

Swap space is a partition on the storage device that is used when the system runs out of physical memory. Linux pushes some of the unused files from RAM to the swap space to free up memory. Usually, the swap space equals twice the RAM capacity.



Swap space on a storage drive.

THE fdisk UTILITY

The **fdisk** utility is a menu-driven program that is used to create, modify, or delete partitions on a storage drive. Using **fdisk**, you can create a new partition table or modify existing entries on the partition table. The **fdisk** utility understands the DOS and Linux type partition tables. The **fdisk** utility also enables you to specify the size of partitions.

```

root@server01:~# fdisk /dev/sda
[Storage device]
WARNING: fdisk GPT support is currently new, and therefore
in an experimental phase. Use at your own discretion.
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help): [Interactive menu prompt]
  
```

The fdisk utility.

SYNTA

X

The syntax of the **fdisk** utility is **fdisk [options] {device name}**

fdisk COMMAND OPTIONS

The **fdisk** utility supports a number of command-line options.

Option	Used To
-b {sector size}	Specify the number of drive sectors.

Option	Used To
-H {heads}	Specify the number of drive heads.
-S {sectors}	Specify the number of sectors per track.
-s {partition}	Print the partition size in blocks.
-l	List partition tables for devices.

fdisk MENU OPTIONS

Aside from supplying command-line options, you can also choose various options when you are working in the **fdisk** menu.

Option	Used To
n	Create a new partition. The sub-options enables you to specify the partition type and partition size.
d	Remove a partition.
p	List the existing partitions.
w	Write the changes to the drive and exit the utility.
q	Cancel the changes made and exit the utility.

```

root@server01:~# fdisk /dev/sda
Command (m for help): p
Disk /dev/sda: 500.1 GB, 500107862016 bytes, 976773168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk label type: gpt
Disk identifier: 25A9390B-9E0A-4336-A55F-D0774B18A3F0

      #   Start     End   Size Type      Name
      1    2048    411647   200M  EFI System  EFI System Partition
      2    411648  2508799     1G  Microsoft basic
      3   2508800  228491263  107.8G Linux LVM
      4   228491264  236879871     4G  Linux filesystem
      5  236879872  251953151     7.2G Microsoft basic

```

Using **fdisk** to list a storage device's partitions.

GNU PARTED

The **GNU Parted** utility is also used to manage partitions. It is particularly useful when creating partitions on new storage drives. It can be used to create, destroy, and resize partitions. The **parted** command runs the GNU Parted utility. Like **fdisk**, **parted** includes a menu-driven interactive mode where you can input various options.

The screenshot shows a terminal window titled 'root@server01:~'. The command entered is '[root@server01 ~]# parted /dev/sda'. The output of the command is displayed, starting with 'GNU Parted 3.1' and 'Using /dev/sda'. A callout box labeled 'Storage device' points to the line 'Using /dev/sda'. Another callout box labeled 'Interactive menu prompt' points to the '(parted)' prompt at the beginning of the output.

The GNU Parted utility.

SYNTA X

The syntax of the **parted** command is **parted [options] {device name}**

GNU PARTED MENU OPTIONS

There are a number of options you can choose when working in GNU Parted's interactive mode.

The screenshot shows a terminal window titled 'root@server01:~'. The command entered is '(parted) print'. A callout box labeled 'Print partitions' points to this command. The output shows details about a disk: Model: ATA ST500LM000-SSHD- (scsi), Disk /dev/sda: 500GB, Sector size (logical/physical): 512B/4096B, Partition Table: gpt, Disk Flags:. Below this, a table lists the partitions:

Number	Start	End	Size	File system	Name
1	1049kB	211MB	210MB	fat16	EFI System Par
2	211MB	1285MB	1074MB	xfs	
3	1285MB	117GB	116GB		

Using GNU Parted to list a storage device's partitions.

The following table lists some of the menu options available.

Option	Used To
select	Choose which device or partition to modify.
mkpart	Create a partition with a specified file system type.
print	List the partition table.

Option	Used To
<code>resizepart</code>	Modify the end position of a partition.
<code>rm</code>	Delete a partition.
<code>quit</code>	Exit the GNU Parted utility.

THE partprobe COMMAND

The **partprobe** command is used to update the kernel with changes in the partition table. The command first checks the partition table, and if there are any changes, it automatically updates the kernel with the changes.

After creating a partition with **fdisk**, you cannot add a file system to that partition unless the kernel can read it from the partition table. You might receive an error like "Re-reading the partition table failed" during the **fdisk** operation. Rebooting the machine updates the table, or you can issue **partprobe** instead to update the table without a reboot.

The **partprobe** command comes packaged with the GNU parted utility.

```

root@server01:~#
The partition table has been altered!
Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 1
6: Device or resource busy.
The kernel still uses the old table. The new table will be
used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
[root@server01 ~]# partprobe ← Update kernel with
[root@server01 ~]# partition changes

```

Updating the kernel after creating a new partition.

SYNTAX

The syntax of the **partprobe** utility is **partprobe [options] [device name]**

THE mkfs COMMAND

The **mkfs** command is used to build a Linux file system on a device, which is usually a drive partition. The following table lists some options of the **mkfs** command and their descriptions.

Option	Used To
<code>-V</code>	Produce verbose output, where the output message will keep changing constantly as the program is processing.

Option	Used To
-V	Produce verbose output, including all file system-specific commands that are executed.
-t {fs type}	Specify the type of file system to be built.
fs-options	Pass file system-specific options to the file system builder.
-c	Check the device for bad blocks before building the file system.
-l {file name}	Read the list of bad blocks from a specified file.

```
root@server01 ~]# mkfs.ext4 /dev/sda6
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
131072 inodes, 524288 blocks
26214 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=536870912
16 block groups
```

Creating a file system on a partition.

SYNTAX

X

One syntax option of the **mkfs** command is **mkfs [options] {device name}**

Another syntax option is **mkfs.{file system type} [options] {device name}**

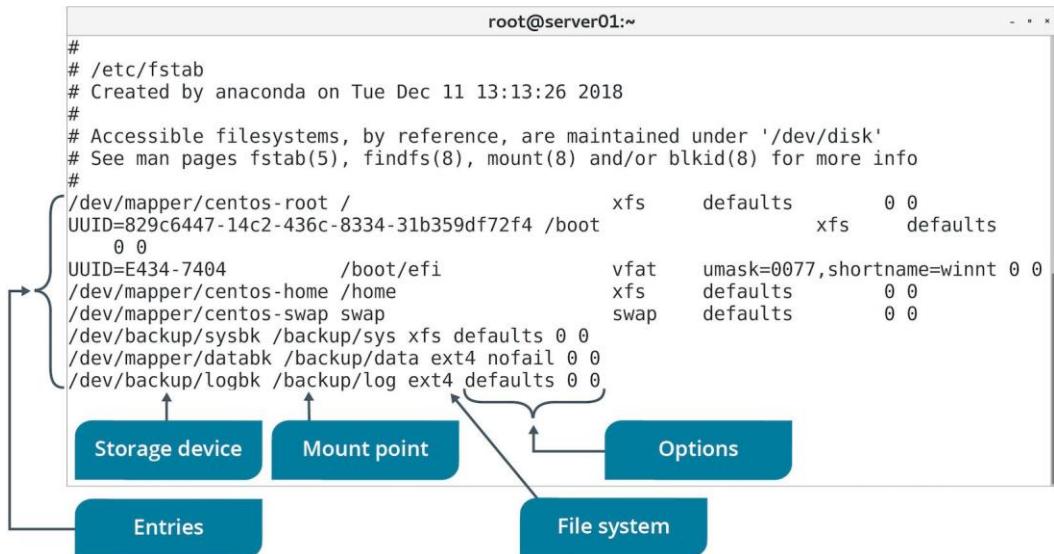
THE **fstab** FILE

The **fstab** file is a configuration file that stores information about storage devices and partitions and where and how the partitions should be mounted. The **fstab** file is located in the **/etc** directory. It is read by the system at boot time and can be edited only by a root user. The **fstab** file consists of a number of lines, one for each file system.

Each line in an **fstab** file has six fields that are separated by spaces or tabs.

Field	Description
Device or partition name	Specifies the name of the device or file system to mount.
Default mount point	Indicates where the file system is to be mounted.
File system type	Specifies the type of file system used by the device or partition.
Mount options	Specifies a set of comma-separated options that will be activated when the file system is mounted.

Field	Description
dump options	Indicates if the dump utility should back up the file system. Usually, zero is specified as the dump option to indicate that dump can ignore the file system.
fsck options	Specifies the order in which the fsck utility should check file systems.

The `fstab` file.

THE crypttab FILE

The `/etc/crypttab` file performs a similar function to the `fstab` file, but its purpose is to store information about encrypted devices and partitions that must be unlocked and mounted on system boot. Its format is similar to `fstab`, but includes an optional password field for unlocking the encrypted device.

THE STORAGE DEVICE SETUP PROCESS

Putting several of the previous tools together, the overall process of setting up a storage device for use on Linux is as follows:

1. Partition the storage device using a tool like `fdisk` or `parted`
2. Format the partition with a file system using the `mkfs` tool.
3. Add the formatted partition to the `fstab` file so that it is configured by the system on boot.

THE `/dev/` DIRECTORY

The `/dev/` directory contains files that represent and support devices attached to the system. Linux uses naming conventions so that storage devices will be easily located by the system when they are attached and when the system boots. For storage devices, the naming convention is typically expressed in three parts. Take the device name `/dev/sda1` as an example:

- The `s` portion refers to a specific type of controller that the device is using (in this case, SCSI/SATA, which is the most common).

- The **a** portion refers to the first whole drive. The second whole drive would be **b**, the third would be **c**, and so on.
- The **1** refers to the first partition on this drive. The second partition would be **2**, the third would be **3**, and so on.

When you manage partitions and other device information, for the most part, you'll use this naming convention to refer to those devices.

THE /dev/disk/by- IDENTIFIERS

In addition to the previous naming convention, Linux also uses several different persistent naming schemes to identify devices. Controller-based naming can become problematic in systems with more than one of the same type of controller. So, the persistent naming schemes were created to make identifying a device more predictable. Some of those schemes are:

- **/dev/disk/by-id**—This refers to an identifier based on the device's hardware serial number.
- **/dev/disk/by-path** —This refers to an identifier based on the shortest physical path to the device (i.e., the path changes if you plug the device into a different port on the controller). This is useful in configurations using DM-Multipath, a feature of the kernel that supports multiple I/O paths to devices.
- **/dev/disk/by-uuid** —This refers to an identifier based on the universally unique identifier (UUID) that was assigned to the device when a file system was created on it.

 **Note:** To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 4-1

Discussing Partitions

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You're a new system administrator and your manager comes to your cubicle to give you a pop quiz over a few concepts. The first one she asks you is to show her how you would set up a new partition on the second storage drive on a Linux system. Which command can you enter to perform this operation?

2. A junior administrator asks for your help with setting up a new Linux file system, /dev/sdb2. He retraces his steps with fdisk, creating the partition and saving the setup. But now, he can't figure out why he can't use the file system—how can you help him?

3. You created a new partition, /dev/sdb1, created the file system (XFS), and mounted the file system as /Files. You also created a few directories on / Files for various groups within your organization as shared file spaces. After a maintenance reboot a few days later, users are complaining that their directories and files that you set up no longer exist. What is the problem?

4. One of the developers has told your manager that he needs the file system / code rebuilt using the XFS file system type. Which utility can you use?

5. You and the other system administrators have a maintenance partition, / maint, that you only want mounted when required for administrative activities. How can you use the file system only when you need it?

Activity 4-2

Creating Partitions

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

Develetech is concerned about losing data in the event of storage drive corruption. You'll create a couple of partitions where the team can store backups for the time being. In the event that the main storage partition fails, these backups may remain unaffected and can be used for recovery.

So, you'll create two new partitions:

- A partition that will hold system data in an XFS file system.
- A partition that will hold critical business files and other user data in an ext4 file system.

You'll just create the partitions for now; you'll make them available for use later.

1. Create the first new logical partition.

- a) Enter **sudo fdisk /dev/sda** to begin partitioning the main drive.

If your computer has multiple physical drives, they will likely be named **/dev/sdb**, **/dev/sdc**, etc.

- b) Enter **M** to view the command menu options.

- c) Enter **P** to print the partition table.

- d) Enter **N** to begin creating a new partition.

- e) Press **Enter** to accept the default partition number.

Several partitions were automatically created during the installation of CentOS, so this is the next available partition number.

- f) Write this partition number down: _____

You'll need to reference the number later.

- g) Press **Enter** to accept the default first sector.

This is the first part of the drive that currently has no partition (i.e., free space).

- h) Enter **+4096M** to create a partition 4 GB in size.

- i) Enter **P** to print the partition table.

- j) Verify that the new partition was created.

#	Start	End	Size	Type	Name
1	2048	411647	200M	EFI System	EFI System Partition
2	411648	2508799	1G	Microsoft basic	
3	2508800	228491263	107.8G	Linux LVM	
4	228491264	236879871	4G	Linux filesystem	

- k) Enter **W** to write your changes to the partition table and exit.



Note: You can ignore the warning; you'll handle this in the next step.

- j) Enter **sudo partprobe** to update the Linux kernel with the partition table changes.
2. Create an XFS file system on the new partition.
 - a) Enter **sudo mkfs.xfs /dev/sda#** where # corresponds to the number of the partition you just created.
 - b) Verify that you are presented with various statistics about the new file system.
 3. Create the second new partition with an ext4 file system.
 - a) Enter **sudo parted /dev/sda**
 - b) Enter **print** to see the list of partitions on this device.
 - c) Enter **mkpart** to start the partition creation process.
 - d) Press **Enter** to select a blank name for now.
 - e) Enter **ext4** as the file system type.



Note: This associates the partition with the ext4 file system type; it doesn't actually format the partition with an ext4 file system.

- f) Examine the list of partitions you just printed and identify the **End** value for the last partition (i.e., the XFS partition you created with **fdisk**).
- g) Enter this value at the **Start?** prompt.
For example, if the end value is **121GB**, then enter **121GB** at the prompt.
- h) At the **End?** prompt, add 8 GB to the start value and enter that.
For example, if the end value is **121GB**, then you'd enter **129GB** at the prompt.
- i) Enter **print** and verify that both of your new partitions appear.

Number	Start	End	Size	File system	Name	Flags	Partition
1	1049kB	211MB	210MB	fat16	EFI System Partition	boot	
2	211MB	1285MB	1074MB	xfs			
3	1285MB	117GB	116GB				lvm
4	117GB	121GB	4295MB	xfs			
5	121GB	129GB	7718MB				

- j) Write down the partition number of the partition you just created:

 - k) Enter **Q** to quit GNU Parted.
4. Create an ext4 file system on the new partition.
 - a) Enter **sudo mkfs.ext4 /dev/sda#** where # corresponds to the number of the partition you just created.
 - b) Verify that you are presented with various statistics about the new file system.
 5. Apply labels to the new partitions.
 - a) Enter **sudo xfs_admin -l /dev/sda#** where # corresponds to the partition number of the XFS partition.
 - b) Verify that the partition currently has no label.
 - c) Enter **sudo xfs_admin -L SysBackup /dev/sda#** to set the label.

- d) Enter **sudo e2label /dev/sda#** where # corresponds to the partition number of the ext4 partition.
 - e) Verify that the partition currently has no label.
 - f) Enter **sudo e2label /dev/sda# DataBackup** to set the label.
 - g) Repeat steps 6a and 6d and verify that both partitions have the labels you set.
-

Topic B

Manage Logical Volumes



EXAM OBJECTIVES COVERED

- 1.4 Given a scenario, manage storage in a Linux environment.
- 4.1 Given a scenario, analyze system properties and remediate accordingly.

Partitions are useful, but they are not the only method for logically dividing a storage device. In this topic, you'll use logical volumes to create a more flexible structure for organizing data on storage devices.

DEVICE MAPPING

Thus far, you've worked with physical storage devices, also called physical volumes. However, there is a way to further abstract these devices into virtual storage devices—a process called **device mapping**. In Linux, the device mapper creates the virtual device and passes data from that virtual device to one or more physical devices.

Several Linux applications leverage the device mapper to perform various tasks. For example, volume encryption and integrity checking services use device mapper to act upon data that is transmitted between physical and virtual devices.

DM-MULTIPATH

DM-Multipath is a feature of the Linux kernel that provides redundancy and improved performance for block storage devices. It leverages the device mapper to support multiple I/O paths (connection interfaces) between the CPU and the storage devices. If one path fails, DM-Multipath will switch to one of the other paths that remain, keeping the storage device available for reading and writing. The **multipath-tools** package enables you to manage DM-Multipath for storage devices, and the typical configuration file is located at `/etc/multipath.conf`

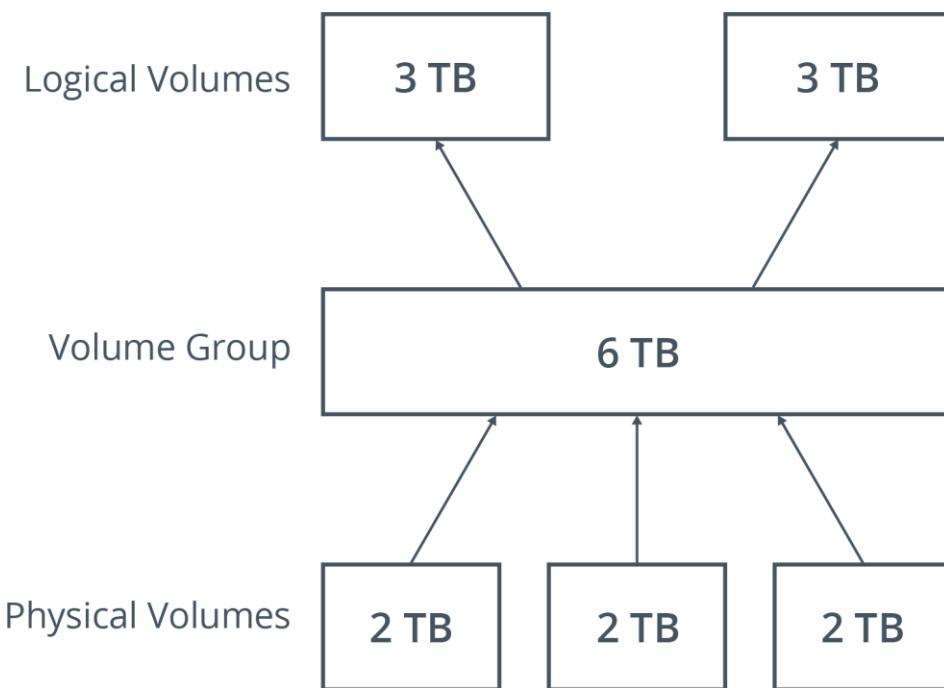
THE mdadm COMMAND

The **mdadm** command is a tool used to manage software-based RAID arrays. **Redundant array of independent disks (RAID)** is a set of vendor-independent specifications that support redundancy and fault tolerance for configurations on multiple-device storage systems. In a RAID array, data is stored across multiple physical storage devices, and those devices are combined into a single virtual storage device. This type of software-based RAID configuration is an alternative to using device mapper and DM-Multipath. The **mdadm** tool enables you to create, manage, and monitor RAID arrays.

LOGICAL VOLUME MANAGER

One major application of the device mapper is the **Logical Volume Manager (LVM)**. LVM maps whole physical devices and partitions (e.g., `/dev/sda1`, `/dev/sdb2`, etc.) into one or more virtual containers called volume groups. Within these volume groups are one or more logical volumes. Ultimately, the logical volumes become the storage devices that the system, user, and applications work with.

Many distributions support LVM, and several actually place the root file system on logical volumes during installation.



An LVM architecture example.

LVM ADVANTAGES

Compared to traditional physical partition management, LVM provides the following benefits:

- You can dynamically create, delete, and resize volumes without having to reboot the system.
- Day-to-day management of volumes is easier once everything is set up.
- You can map multiple logical volumes across multiple physical devices.
- A logical volume can exceed the size of any one physical device, as long as it doesn't exceed the total size of devices in the volume group.
- You can create virtual snapshots of each logical volume so you can quickly and easily revert a volume to a specific state.

One potential downside to LVM is that the initial setup can be somewhat complex.

THE /dev/mapper/ DIRECTORY

The `/dev/mapper/` directory contains all of the logical volumes on the system that are managed by LVM. Devices in this directory are typically formatted as:

`/dev/mapper/<volume group name>-<logical volume name>`

In some cases, this directory may just include links to the actual logical volume location.



Note: Logical volumes may also be in the path `/dev/<volume group name>/<logical volume name>`

LVM TOOLS

LVM divides its volume management tools into three categories based on the three different components that make up LVM:

- Physical volume (PV) tools
- Volume group (VG) tools
- Logical volume (LV) tools

PV TOOLS

The following table lists some of LVM's physical volume (PV) tools.

Tool	Used To
pvscan	Scan for all physical devices that are being used as physical volumes.
pvcreate	Initialize a drive or partition to use as a physical volume.
pvdisplay	List attributes of physical volumes.
pvchange	Change attributes of a physical volume.
pvs	Display information about physical volumes.
pvck	Check the metadata of physical volumes.
pvremove	Remove physical volumes.

VG TOOLS

The following table lists some of LVM's volume group (VG) tools.

Tool	Used To
vgscan	Scan all physical devices for volume groups.
vgcreate	Create volume groups.
vgdisplay	List attributes of volume groups.
vgchange	Change attributes of volume groups.
vgs	Display information about volume groups.
vgck	Check the metadata of volume groups.
vgrename	Rename a volume group.
vgreduce	Remove physical volumes from a volume group to reduce its size.
vgextend	Add physical volumes to volume groups.
vgmerge	Merge two volume groups.
vgsplit	Split a volume group into two.
vgremove	Remove volume groups.

LV TOOLS

The following table lists some of LVM's logical volume (LV) tools.

Tool	Used To
lvscan	Scan all physical devices for logical volumes.
lvcreate	Create logical volumes in a volume group.
lvdisplay	List attributes of logical volumes.
lvchange	Change attributes of logical volumes.

Tool	Used To
lvs	Display information about logical volumes.
lvrename	Rename logical volumes.
lvreduce	Reduce the size of logical volumes.
lvextend	Extend the size of logical volumes.
lvresize	Resize logical volumes.
lvremove	Remove logical volumes.



Note: By default, the argument you provide to any of the resizing commands resizes the total LV to that value. You can also use + and – to resize by offset values instead.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 4-3

Discussing Logical Volumes

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **You need to create a new volume for Human Resources. That group has purchased a new 1 TB drive for their volume. What is the first step in the process of creating the logical volume?**

 2. **How can you create a physical volume on /dev/sdd1?**

 3. **How can you remove the physical volume /dev/sde1?**

 4. **Which command can you use to create a new volume group, shared, with /dev/sdb1 and /dev/sdd1? And then how do you view your volume groups?**

 5. **Which command can you use to create a new 400 GB logical volume, finance, from volume group vg_finance?**
-

Activity 4-4

Managing Logical Volumes

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created two backup partitions.

SCENARIO

The two backup partitions you created are acceptable, but you know there are more efficient ways of managing separate storage devices. So, you've decided to consolidate the space in both of these partitions to create a physical volume using the Logical Volume Manager (LVM). You'll create a single volume group for backups that extends across these physical volumes. Then, you'll create logical volumes for both system files and business data files.

As you create these volumes, a colleague tells you that the backup system will need to keep its own set of logs that other members of the team can audit to ensure the backup process is running as expected. He suggests creating another partition for these log files, with the understanding that the size of this partition (and of the other partitions) might need to change over time. You offer to create a logical volume instead, explaining that it'll be much easier to manage than a physical partition. He agrees, so you'll get to work.

1. Identify the current logical volumes on the system.
 - a) Enter **ls /dev/mapper**
 - b) Verify that there are several mapped logical volumes on the system already.
CentOS creates these logical volumes during installation, including **centos-home**, **centos-root**, and **centos-swap**.

2. Scan the system for physical volumes, volume groups, and logical volumes.
 - a) Enter **sudo pvscan**
 - b) Verify that at least one physical device is supporting the logical volumes.

```
[student01@localhost permissions-demo]$ sudo pvscan
PV /dev/sda3    VG centos          lvm2 [107.75 GiB / 4.00 MiB free]
Total: 1 [107.75 GiB] / in use: 1 [107.75 GiB] / in no VG: 0 [0 ]
```
 - c) Enter **sudo pvdisk** to see more details about the physical volume.
The physical volume has a name in the format **/dev/sda#** as well as a volume group name (**centos**) and size.
 - d) Enter **Sudo vgscan** and verify that the **centos** volume group was found.
 - e) Enter **Sudo vgdisplay centos** to see more information about this volume group.
 - f) Enter **Sudo lvscan** and verify that the three CentOS logical volumes were identified.

- g) Enter **sudo lvdisplay /dev/centos/home** to see more information about this particular logical volume.
- 3.** Create physical volumes from the backup partitions you created earlier.
- Enter **sudo pvcreate /dev/sda# /dev/sda#** where # corresponds to each new partition you created previously.
For example: **sudo pvcreate /dev/sda4 /dev/sda5**
This creates a physical volume for each partition.
 - Enter **y** to clear any file system signatures.
 - Enter **sudo pvdisk** and verify that your new physical volumes appear.
- 4.** Create a volume group from these new physical volumes.
- Enter **sudo vgcreate backup /dev/sda# /dev/sda#** where # corresponds to each new partition you created previously.
For example: **sudo vgcreate backup /dev/sda4 /dev/sda5**
This creates a new volume group named **backup** that extends across both physical volumes.
 - Enter **sudo vgdisplay backup** and verify that you can see details about your new volume group.
The total size of the group should be around 11 GB.
- 5.** Create three logical volumes in the new volume group.
- Enter **sudo lvcreate --name sysbk --size 1GB backup**
 - Verify that the logical volume was created.
This also places the logical volume within the **backup** volume group you just created.
 - Enter **sudo lvcreate --name databk --size 2GB backup**
 - Enter **sudo lvcreate --name logbk --size 3GB backup**
 - Enter **sudo lvdisplay backup** and verify that your new logical volumes are listed.
- 6.** Extend the data backup volume and reduce the log volume.
- Enter **sudo lvextend -L5G /dev/backup/databk**
This extends the **databk** volume from 2 GB to 5 GB in size.
 - Enter **sudo lvreduce -L1G /dev/backup/logbk**
 - Enter **y** to accept the warning.
This reduces the **logbk** volume from 3 GB to 1 GB in size.
 - Enter **sudo vgdisplay backup** and verify that the volume group has approximately 4 GB of free space left.



Note: It's a good idea to leave free space in a volume group in case you later need to extend a volume.

- 7.** Create file systems on the logical volumes.
- Enter **sudo mkfs.xfs /dev/backup/sysbk**
Although you created file systems on the partitions earlier, you'll need to create them on the logical volumes as well.
 - Enter **sudo mkfs.ext4 /dev/backup/databk**

- c) Enter **sudo mkfs.ext4 /dev/backup/logbk**
-

Topic C

Mount File Systems



EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

After formatting your partitions and logical volumes with file systems, you need to actually make it possible for users and the system to work with the file systems. So, in this topic, you'll make these file systems available by mounting them.

MOUNT POINTS

A **mount point** is an access point to information stored on a local or remote storage device. The mount point is typically an empty directory on which a file system is loaded, or mounted, to make the file system accessible to users. If the directory already has content, the content becomes invisible to the users until the mounted file system is unmounted.



Note: You can use the *fstab* file to list the file system to be mounted and unmounted when the Linux system boots and shuts down, respectively.

THE mount COMMAND

The **mount** command loads a file system to a specified directory so that it can be accessible to users and applications. You must specify the device to mount as well as the desired mount point.



Mounting partitions on a storage device to specific directories.

SYNTAX

The syntax of the **mount** command is **mount [options] {device name} {mount point}**

mount OPTIONS

You can specify various **mount** options for a file system. These options are typically included in the **fstab** file rather than as command-line arguments.

Option	Used To
auto	Specify that the device has to be mounted automatically.
noauto	Specify that the device should not be mounted automatically.
nouser	Specify that only the root user can mount a device or a file system.
user	Specify that all users can mount a device or a file system.
exec	Allow binaries in a file system to be executed. noexec
	Prevent binaries in a file system from being executed. ro
	Mount a file system as read-only.
rw	Mount a file system with read and write permissions.
sync	Specify that input and output operations in a file system should be done synchronously.
async	Specify that input and output operations in a file system should be done asynchronously.

BINARIES

A binary is source code that is compiled into an executable program, or otherwise assembled so that it is readable by the computer system. In addition, binaries can be pictures, word processing files, or spreadsheet files.

THE umount COMMAND

After using the mounted file system, it can be disassociated from the directory by unloading, or unmounting, the file system using the **umount** command. In order to unmount a file system, it must not be in use—for example, if a file on that file system is currently open in some application.

SYNTAX

The syntax of the **umount** command is **umount [options] {mount point}**

umount COMMAND OPTIONS

Some common command options for **umount** are described in the following table.

Option	Used To
-f	Force a file system to be unmounted despite any detected issues.
-l	Perform a "lazy" unmount, in which the file system is detached from the hierarchy, but references to that file system are not cleaned up until the file system is no longer being used.

Option	Used To
-R	Recursively unmount the specified directory mount points.
-t {fs type}	Unmount only the file system types specified.
-O {mount options}	Unmount only the filesystems that are mounted with the specified options in the /etc/fstab file.
--fake	Test the unmounting procedure without actually performing it.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 4-5

Discussing Mounting File Systems

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **What are the three requirements for mounting a file system?**

2. **What final task do you need to complete in mounting a new file system to have that new file system mount at boot time?**

3. **What is the generic format of the /etc/fstab file?**

4. **How can you unmount the /finance file system?**

5. **If you issue the following command: sudo umount /share1 and you receive the error that the resource is busy, how do you fix the problem and unmount the file system?**

Activity 4-6

Mounting File Systems

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created several logical volumes in the `backup` volume group.

SCENARIO

Your volumes are all in place, but they aren't usable yet. You need to mount them first so that users and applications can access their file systems. You decide to mount each of the three volumes in the `/backup` directory, like this:

- `/backup/sys/`
- `/backup/data/`
- `/backup/log/`

In addition, these volumes need to be mounted automatically in case the system needs to reboot.

1. Create mount points for the logical volumes.
 - a) Enter `sudo mkdir -p /backup/sys /backup/data /backup/log`
 - b) Enter `ls /backup` and verify that the new directories were created.
These will serve as the mount points for your logical volumes.
2. Mount the logical volumes.
 - a) Enter `sudo mount /dev/backup/sysbk /backup/sys`
This mounts the system backup logical volume onto the mount point you just created.
 - b) Enter `sudo mount /dev/backup/databk /backup/data`
 - c) Enter `sudo mount /dev/backup/logbk /backup/log`
 - d) Enter `mount` and verify your backup volumes are mounted.
They'll likely be at the bottom of the list as `/dev/mapper/backup-<LV name>`
3. Unmount a volume.
 - a) Enter `sudo umount /backup/log` to unmount the volume.
 - b) Enter `mount` and verify that the log volume is no longer mounted.
In this state, the volume is not currently usable. You may need to unmount a volume/partition if you'd like to move its mount point. Also, some operations require the file system to be unmounted before they can work with it.
4. Ensure the logical volumes are mounted on boot.
 - a) Enter `sudo vim /etc/fstab`

- b) Press **i** to switch to insert mode.
- c) Use the arrow keys to move down to the end of the last line.
- d) Press **Enter** to start a new line.
- e) On the next three lines, type the following:

```
/dev/backup/sysbk /backup/sys xfs defaults 0 0  
/dev/backup/databk /backup/data ext4 defaults 0 0  
/dev/backup/logbk /backup/log ext4 defaults 0 0
```



Note: It's OK if the lines don't align exactly.

- f) Press **Esc**, then enter **:WQ** to save the file and quit.
- g) Enter **Sudo mount -a** and verify that there are no errors.

This tests the **fstab** file to ensure it can mount all of the volumes that are listed in it. A misconfigured **fstab** file may prevent the system from booting.

Topic D

Manage File Systems



EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

2.7 Explain the use and operation of Linux devices.

4.1 Given a scenario, analyze system properties and remediate accordingly.

Linux provides you with many tools for modifying file systems after applying them to storage devices. In this topic, you'll perform various tasks to ensure your file systems are fulfilling your business needs.

THE /proc/mounts FILE

The `/proc/mounts` file lists the status of all currently mounted file systems in a format similar to `fstab`: the system's name, mount point, file system type, etc. It is actually not a real file, but part of the virtual file system that represents the status of mounted objects as reported by the Linux kernel. Typically, you'd read this "file" using a command like `cat` in order to get the details you're looking for.

Note that `/proc/mounts` lists all file systems, not just those on storage drives and partitions. It may not be the most readable tool to use in case you're only looking for drive information.

THE mtab FILE

The `/etc/mtab` file is very similar to the `/proc/mounts` file in that it reports the status of currently mounted file systems. However, `/proc/mounts` is typically more accurate and includes more up-to-date information about the file systems.

THE /proc/partitions FILE

The `/proc/partitions` file contains information about each partition that is currently attached to the system. Like `/proc/mounts`, it is not a real file but part of the virtual file system. The format of `/proc/partitions` contains columns, and each column is as follows:

- **major**—Represents the class of device so that it can be mapped to an appropriate driver.
- **minor**—Separates partitions into physical devices. This corresponds to the number at the end of the partition's name.
- **#blocks**—How many physical blocks the partition takes up.
- **name**—The name of the partition.

```
root@server01 ~]# cat /proc/partitions
major minor #blocks name

 8       0   488386584 sda
 8       1    204800 sda1
 8       2   1048576 sda2
 8       3 112991232 sda3
 8       4   4194304 sda4
 8       5   7536640 sda5
 8       6   2097152 sda6
11       0   4168560 sr0
253      0  52428800 dm-0 }
```

Partitions

The /proc/partitions file.

THE lsblk COMMAND

The **lsblk** command displays information about all block storage devices that are currently available on the system. The output is displayed in a tree-like format with each physical device at the top of the tree and each partition or logical volume branching off from that device. The information displayed includes names, major and minor numbers, size, device type, and mount point.

```
root@server01 ~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda        8:0     0 465.8G 0 disk
└─sda1     8:1     0  200M 0 part /boot/efi
└─sda2     8:2     0   1G 0 part /boot
└─sda3     8:3     0 107.8G 0 part
  ├─centos-root 253:0   0   50G 0 lvm /
  ├─centos-swap 253:1   0   7.8G 0 lvm [SWAP]
  ├─centos-home 253:2   0   50G 0 lvm /home
  └─sda4     8:4     0   4G 0 part
  └─backup-sysbk 253:3   0   2G 0 lvm /backup/sy
  └─backup-databk 253:4   0   6G 0 lvm
└─sda5     8:5     0   7.2G 0 part
  └─backup-sysbk 253:3   0   2G 0 lvm /backup/sy
```

Block storage devices

Listing block storage devices in a tree-like hierarchy.

SYNTA

X

The syntax of the **lsblk** command is **lsblk [options] [device name]**

lsblk COMMAND OPTIONS

The following table lists some of the options available with the **lsblk** command.

Option	Used To
-a	List empty devices as well.
-e {device list}	Exclude devices from output that you provide as a list of comma-separated major device numbers.
-f	Output additional file system information.
-l	Output results in list format.
-m	Output permissions information for devices.

THE blkid COMMAND

The **blkid** command offers similar functionality to **lsblk**, but it simply prints each block device in a flat format and includes some additional information like device/ partition UUID and file system type. However, it is preferable to use **lsblk -f** if you want this additional information.

SYNTAX

The syntax of the **blkid** command is **blkid [options] [device name]**

ext TOOLS

When it comes to managing file systems and partitions, some tools are designed to only work with specific file system types. Some can work with multiple types, but are much better suited to one or a few specific types. When it comes to ext file systems, there are several tools that you can use on essentially any generation of the ext type (i.e., ext2/3/4).

Some of the most common and useful tools for managing ext file systems include:

- **e2fsck**
- **resize2fs**
- **tune2fs**
- **dumpe2fs**

THE fsck COMMAND

The **fsck** command is used to check the integrity of a file system. **File system integrity** refers to the correctness and validity of a file system. Most systems automatically run the **fsck** command at boot time so that errors, if any, are detected and corrected before the system is used. File system errors are usually caused by power failures, hardware failures, or improper shutdown of the system.

You should unmount the file system before scanning it with **fsck** to prevent damage to the file system.



Note: The *fsck* command is essentially a front-end that calls the appropriate tool for the type of file system you're working with. For ext file systems, it will call the *e2fsck* tool.

Specific tool called

Partition with file system

No errors detected

```
root@server01 ~]# fsck /dev/sda6
fsck from util-linux 2.23.2
e2fsck 1.42.9 (28-Dec-2013)
/dev/sda6: clean, 11/131072 files, 26156/
524288 blocks
```

Checking the integrity of a file system.

SYNTAX

X

The syntax of the **fsck** command is **fsck [options] {device/file system name}**

REPAIR FILE SYSTEMS

You can use the **fsck -r {device/file system name}** command to repair a file system. The command will prompt you to confirm your actions. If you are simultaneously checking multiple file systems, you should not use this option because it allows you to repair only a single file system at a time.

THE **resize2fs** COMMAND

The **resize2fs** command is used to enlarge or shrink an ext2/3/4 file system on a device. You can enlarge a mounted file system, but you must unmount the file system before you can shrink it. You can specify the desired size of the file system in order to either enlarge or shrink it. If you don't specify a size, the file system will be resized to the same size as the partition.

It's important to note that **resize2fs** does not resize partitions, only the file system. You must use a command like **fdisk** or an LVM tool to expand the size of the partition/ volume first in order to then enlarge the file system.

SYNTAX

The syntax of the **resize2fs** command is **resize2fs [options] {device/file system name} [desired size]**

THE **tune2fs** COMMAND

The **tune2fs** command helps you configure various "tunable" parameters associated with an ext2/3/4 file system. Tunable parameters enable you to remove reserved blocks, alter reserved block count, specify the number of mounts between checks, specify the time interval between checks, and more.

You can also use **tune2fs** to add a journal to an existing ext2 or ext3 file system (neither of which include journaling by default). If the file system is already mounted,

the journal will be visible in the root directory of the file system. If the file system is not mounted, the journal will be hidden.

SYNTAX

The syntax of the **tune2fs** command is **tune2fs [options] {device/file system name}**

tune2fs COMMAND OPTIONS

The **tune2fs** command has various options.

Option	Used To
-j	Add an ext3 journal to the existing file system.
-i {d m w}	Specify the maximum time interval between file system checks in days, months, or weeks.
-c {maximum mounts count}	Specify the maximum number of mounts between file system checks.
-C {mount count}	Specify the number of times the file system can be mounted.
-r {reserved blocks count}	Specify the number of reserved file system blocks.
-e {continue remount-ro panic}	Specify the behavior of the kernel code, whether the file system should continue with normal execution, remount the file system in read-only mode, or cause a kernel panic, when errors are detected.
-l	List the contents within the superblock (metadata) of the file system.
-U {UUID}	Set the specified UUID for the file system.

SUPERBLOCK

A file system's **superblock** contains metadata about that file system, including its size, type, and status. The superblock is critical to the function of the file system, and if it becomes corrupt, you may be unable to mount and work with the file system. You can use a tool like **fsck** to repair the superblock, if necessary.

THE **dumpe2fs** COMMAND

The **dumpe2fs** command is used to dump ext2, ext3, and ext4 file system information. It prints the superblock and block group information for the selected device. This can be useful when troubleshooting a faulty file system.

File system info

```
[root@server01 ~]# dumpe2fs /dev/sda6
dumpe2fs 1.42.9 (28-Dec-2013)
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: 23f3d1ef-35ad-4db1-a157-79171bbbe9fb
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype extent 64bit flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags: signed_directory_hash
Default mount options: user_xattr acl
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 131072
Block count: 524288
```

Dumping file system information.

SYNTAX

X

The syntax of the **dumpe2fs** command is **dumpe2fs [options] {device/file system name}**

dumpe2fs COMMAND OPTIONS

The **dumpe2fs** command has various options.

Option	Used To
-x	Print a detailed report about block numbers in the file system.
-b	Print the bad blocks in the file system.
-f	Force the utility to display the file system status irrespective of the file system flags.
-i	Display file system data from an image file created using the e2image command.

XFS TOOLS

There are many tools that enable you to work with the XFS file system. The following table lists some of those tools.

Tool	Used To
xfs_info	Display details about the XFS file system, including its block information.
xfs_admin	Change the parameters of an XFS file system, including its label and UUID.
xfs_metadump	Copy the superblock metadata of the XFS file system to a file.
xfs_growfs	Expand the XFS file system to fill the drive size.
xfs_copy	Copy the contents of the XFS file system to another location.
xfs_repair	Repair and recover a corrupt XFS file system.

Tool	Used To
xfs_db	Debug the XFS file system.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 4-7

Discussing Managing File Systems

SCENARIO

Answer the following questions to check your understanding of the topic.

-
1. **What are the steps required to grow the file system /dev/data/ datastore1 from its current 200 GB size to 400 GB?**
 2. **Why should you have to unmount a file system prior to performing an fsck on it?**
 3. **You know that for ext4 file systems, you use the resizefs utility to grow the file system, but which utility can you use to perform this same action on XFS file systems?**
 4. **Another system administrator appears in your cubicle and asks you to display storage devices and their file systems on server05. Which command can you use to show him the info he needs?**
 5. **You have a coworker who attempted to grow a file system on a Linux server. His problem is that the file system didn't extend despite entering the following command: sudo lvextend -L150G /dev/shared/ graphics—He remounted the file system onto /Graphics but the new space doesn't show up. What is the issue?**

Activity 4-8

Managing File Systems

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

So far, your new volumes have been working great. However, because backups are critical to the business, you want to perform scheduled maintenance on each file system to spot any errors that could corrupt data or make it inaccessible.

Also, as expected, the scope of your backup operations has changed, and it's become necessary to expand the size of your data backup and log backup volumes to accommodate these changes. You'll need to ensure that you resize their file systems as well, or users and apps won't be able to avail themselves of the newly added space.

1. Query the system for information about block storage devices.
 - a) Enter **lsblk -f**
 - b) Verify that each device, partition, and volume is listed, along with file system information.
2. **Examine the tree structure. What can you identify about your storage devices?**
3. Scan two of the logical volumes for errors.
 - a) Enter **sudo umount /dev/backup/databk**
A volume must be unmounted before it can be checked for errors.
 - b) Enter **sudo fsck /dev/backup/databk**
 - c) Verify that the utility is reporting the volume as clean, indicating there are no detected errors. If any were to be detected, you could run **sudo fsck -r** to repair them.
 - d) Enter **sudo umount /dev/backup/sysbk** to unmount the XFS file system.
 - e) Enter **sudo xfs_repair /dev/backup/sysbk**
 - f) Verify that the utility attempted to identify and repair any errors. If any were to be identified, they would be fixed automatically.
4. Increase the size of an ext4 file system on a volume.

- a) Enter **sudo e2fsck -f /dev/backup/databk** and make note of the total amount of blocks on the file system:

 - b) Enter **sudo lvextend -L6G /dev/backup/databk**
This extends the **databk** volume from 5 GB to 6 GB in size.
 - c) Enter **sudo e2fsck -f /dev/backup/databk**
Despite increasing the size of the volume, the file system did not increase in size with it.
 - d) Enter **sudo resize2fs /dev/backup/databk**
 - e) Verify that the file system was resized, and that there are more total blocks than before.
 - f) Enter **sudo mount /dev/backup/databk /backup/data** to remount the ext4 file system.
5. Increase the size of an XFS file system on a volume.
- a) Enter **sudo lvextend -L2G /dev/backup/sysbk**
 - b) Enter **sudo mount /dev/backup/sysbk /backup/sys** to remount the XFS file system.
The following command requires the file system to be mounted.
 - c) Enter **sudo xfs_growfs /dev/backup/sysbk**
Verify that the total number of data blocks increased on the file system.
-

Topic E

Navigate the Linux Directory Structure



EXAM OBJECTIVES COVERED

1.4 Given a scenario, manage storage in a Linux environment.

Now that you've managed storage at a lower level, you can begin to explore the standard file system structure that applies to most Linux distributions. You'll be navigating this file structure all throughout this course, as well as in your daily duties when you're on the job.

TYPES OF FILES

Linux contains regular files that include text files, executable files or programs, input for programs, and output from programs. Besides these, the Linux file system consists of other types of files, as described in the following table.

File	Description
Directories (d)	A container for other files.
Special files	Includes system files. These files are stored in the /dev directory. They can be block special files (b) or character special files (c). Block special files are large files that are used for data storage. Character special files are small files that are used for streaming of data.
Links (l)	Makes a file accessible in multiple parts of the system's file tree.
Domain sockets (s)	Provides inter-process networking that is protected by the file system's access control.
Named pipes (p)	Enables processes to communicate with each other without using network sockets.



Note: The individual letters correspond to how each file type is represented in the first bit of the `ls -l` command.

THE `file` COMMAND

The `file` command is used to determine the type of file. The syntax of the command is `file [options] {file names}`

FILE NAMING CONVENTIONS

A file name is a string of characters that identify a file. By using the right combination of characters in file names, you can ensure that the files are unique and easy to recognize.

On an ext4 file system, a file name may be up to 255 bytes long and contain any byte except NULL (\0) and the forward slash (/). File names of user files may not be -.

and .. as these are special reserved file names. Various file systems may enforce different requirements for file names.

Although file names may contain a space, convention on Linux systems dictates that words in a file name are more frequently demarcated by a hyphen or an underscore, as these are easier to manage on the command-line. For example: **audit- file.txt** or **audit_file.txt** are acceptable.

THE FILESYSTEM HIERARCHY STANDARD

The **Filesystem Hierarchy Standard (FHS)** is a collaborative document that specifies a set of guidelines for the names of files and directories and their locations on Linux systems. Most Linux distributions are FHS-compliant, and therefore support compatibility with other systems. The FHS also creates a naming convention that helps administrators, users, and applications consistently find the files they are looking for, as well as store files where other entities can easily find them.



Note: The complete documentation for FHS is available at www.pathname.com/fhs/.

STANDARD DIRECTORIES

As defined in the FHS, the top-most directory in a Linux file system is the root directory, indicated by a single forward slash (/). Below the root directory are various subdirectories that are standardized as part of the FHS. The following table describes these directories in alphabetical order.

Director	Description
/bin	Stores essential command-line utilities and binaries. For example, the /bin/ls is the binary for the ls command.
/boot	Stores the files necessary to boot the Linux operating system.
/dev	Stores hardware and software device drivers. This directory maintains file system entries that represent the devices connected to the system. For example, the /dev/sda1 partition.
/etc	Stores basic configuration files. For example, the /etc/samba/smb.conf file stores Samba configuration data.
/home	Stores users' home directories, including personal files.
/lib	Stores shared program libraries required by the kernel, command-line utilities, and binaries.
/media	Stores mount points for removable media such as CD-ROMs and floppy disks.
/mnt	This is the mount point for temporarily mounting file systems.
/opt	Stores optional files of large software packages. These packages normally create a subdirectory bearing their name under the /opt directory and then place their files in the subdirectory. For example, the /opt/nessus subdirectory contains files for the Nessus vulnerability scanning program.
/proc	This is a virtual file system (VFS) that represents continually updated kernel information to the user in a typical file format. For example, the /proc/mounts file.

Director	Description
/root	The home directory of the root user.
/sbin	Stores binaries that are used for completing the booting process and also the ones that are used by the root user. For example, the /sbin/ifconfig file is the binary for the ifconfig command that is used to manage network interfaces on the system.
/sys	This is another VFS, and it primarily stores information about devices. For example, /sys/block includes links to devices that are stored in various subdirectories under the /sys/devices location, which presents a hierarchy of devices in the kernel.
/tmp	Stores temporary files that may be lost on system shutdown.
/usr	A read-only directory that stores small programs and files accessible to all users.
/var	Stores variable files, or files that are expected to constantly change as the system runs. Examples include log files, printer spools, and some networking services' configuration files.

/usr SUBDIRECTORIES

The **/usr** directory contains some important subdirectories.

Subdirectory	Description
/usr/bin	Includes executable programs that can be executed by all users.
/usr/local	Includes custom build applications that are stored here by default.
/usr/lib	Includes object libraries and internal binaries that are needed by the executable programs.
/usr/lib64	Serves the same purpose as /usr/lib , except that it is meant only for 64-bit systems.
/usr/share	Includes read-only architecture independent files. These files can be shared among different architectures of an operating system.

THE HOME DIRECTORY

The **home directory** contains a user's personal files or files that are otherwise specific to that user. The home directory is where you are placed when you log in to the system. In Linux, by default, every user except the root user is assigned a subdirectory in **/home** that corresponds to their user name. A user can create subdirectories and files within this directory.

The home directory of the root user is **/root**. This is not to be confused with the root directory (**/**), which is the top-most part of the file system hierarchy.

In many shells, including KornShell, C shell, and Bash, the tilde character (**~**) represents your home directory.

```
[root@server01 ~]# ls -al
total 76
dr-xr-x---. 18 root root 4096 Jan  7 15:12 .
dr-xr-xr-x. 21 root root 4096 Jan  2 18:41 ..
-rw-----.  1 root root 1688 Dec 11 18:28 anaconda-ks.cfg
-rw-----.  1 root root 2823 Jan  2 18:08 .bash_history
-rw-r--r--.  1 root root   18 Dec 29 2013 .bash_logout
-rw-r--r--.  1 root root  176 Dec 29 2013 .bash_profile
-rw-r--r--.  1 root root  176 Dec 29 2013 .bashrc
drwx-----. 17 root root 4096 Jan  2 14:56 .cache
drwx-----. 16 root root 4096 Jan  2 18:04 .config
-rw-r--r--.  1 root root  100 Dec 29 2013 .cshrc
drwx-----.  3 root root   25 Dec 11 18:31 .dbus
drwxr-xr-x.  2 root root    6 Dec 11 18:34 Desktop
```

The contents of the root user's home directory.

THE CURRENT WORKING DIRECTORY

The **current working directory (CWD)** is the location on the system that you are accessing at any point in time. For example, when you log in to a system, you are placed in your home directory. So, your current working directory is your home directory. The current working directory is represented in shorthand as a single period (·).

Remember, you can enter `pwd` to identify your current working directory.

THE PARENT DIRECTORY

The **parent directory** is one level above your current working directory. All directories, except the root directory, have a parent directory. You can use the double period notation (··) to switch to the parent directory.

PATHS

A **path** specifies a location in the file system. It begins with the root directory, the directory at the top of the directory tree, and ends with the directory or file you want to access. Thus far, you've worked with paths in order to access various files and directories.

You can refer to a particular file by providing a path to the specific directory that contains the file. For example, the home directory `jsmith` contains a subdirectory, `work`, which contains a file named `mywork`. To refer to that file, use the following path name: `/home/jsmith/work/mywork`. Notice that the forward slash (/) character is used to separate items in the path. The slash that precedes `home` represents the root directory, from which the path to the file `mywork` begins.

ABSOLUTE VS. RELATIVE PATHS

Paths are of two types—absolute and relative. An **absolute path** refers to the specific location irrespective of the current working directory or combined paths. These paths are usually written with reference to the root directory, and, therefore, start with a

forward slash. Paths that do not begin with a forward slash are called relative paths. A **relative path** is the path relative to the current working directory; therefore, the full absolute path need not be included. These paths can contain the period (.) and double period (..), which are indications for the current and parent directories.

EXAMPLES

The following examples show accessing the same location using absolute and relative paths, respectively.

Using an absolute path:

```
cd /usr/bin
```

Using a relative path when the CWD is /usr/:

```
cd bin
```

FILE SYSTEM NAVIGATION COMMANDS

The following table is a review of file navigation commands you've seen thus far.

Command	Used To
cd	Traverse the directory structure using absolute or relative paths to change your current working directory.
ls	List the files and directories in the current working directory or the relative/absolute path you specify. In long listing format (–l), the first bit indicates the type of file.
pwd	Print the current working directory to the console.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 4-9

Discussing the Linux Directory Structure

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **If you cd to the root directory of the file system, name three directories that you'll see there.**

2. **What standard directory contains system logs?**

3. **How can you list files in a hidden directory?**

4. **Explain the difference between /root and /**

5. **If you cd into /opt/log/first/test/test1 and then type cd ~ and press Enter, which directory is now your current working directory?**

Activity 4-10

Navigating the Linux Directory Structure

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You need to become more familiar with the structure and design of a typical Linux file system from a user perspective. That way, you'll be able to easily navigate the file system and find the data and system files that you're looking for at any given time.

You'll walk through several of the main directories that were created during installation.

1. Navigate to a couple of home directories.
 - a) Enter **cd ~**
The **~** symbol is a shortcut to the current user's home directory.
 - b) Enter **pwd** and verify that the current working directory is **/home/student##**
 - c) Enter **SU - root**
 - d) Enter the password.
 - e) Verify that you're logged in as the root account.
 - f) Enter **pwd** and verify that the current working directory is **/root**
Each user has their own home directory within **/home** except for the root user.
 - g) Enter **exit** to log out of the root account and return to your student account.

2. Navigate through **/etc** using relative paths.
 - a) Enter **cd /etc** to change your current working directory.
 - b) Enter **ls -l** and verify that there are several files and subdirectories in this directory.
The **/etc** directory contains configuration files for many services and applications.
 - c) Enter **cd ssh**
 - d) Enter **pwd** and verify that you are now in **/etc/ssh**
You referenced a relative path, as the **ssh** directory is within **/etc** rather than at the root of the file system.
 - e) Enter **cd etc** and verify that no such file or directory was found.

3. **Why was no file or directory found, when you were just in a directory with this name?**

4. Perform some additional navigation using relative and absolute paths.
- Enter `cd ..`
 - Enter `pwd` and verify that you are back in `/etc`
 - Enter `cd /etc/ssh`
- This time, you used an absolute path to return to the `/etc/ssh` directory.

5. Get an overview of the system file structure.
- Enter `cd /` to return to the root of the file system.
 - Enter `ls -l` to get an overview of all of the main directories in the Linux file structure.

6. What type of file object is `bin`?

- Regular directory
- Special file
- Link
- Named pipe

7. Navigate through the Linux file structure as desired to help you answer the following questions.

The `/bin` directory typically contains which of the following files?

- Files marked for deletion.
- Essential command-line utilities.
- Shared program libraries.
- Optional software package files.

Which of the following directories is a virtual file system (VFS) that represents system information reported by the kernel?

- `/usr`
- `/proc`
- `/lib`
- `/opt`

You've installed a third-party software package called `MyApp`. Where is the most likely place that this package is stored?

- `/opt/myapp`
- `/usr/myapp`
- `/sys/myapp`
- `/bin/myapp`

Topic F

Troubleshoot Storage Issues



EXAM OBJECTIVES COVERED

- 1.4 Given a scenario, manage storage in a Linux environment.
- 2.2 Given a scenario, manage users and groups.
- 4.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

Storage issues are quite common when working with any operating system, Linux included. In this topic, you'll do some investigative work to diagnose and suggest solutions for the many issues that can affect storage.

COMMON STORAGE ISSUES

Administrators can face a variety of issues with local storage devices. A key aspect of server management includes ensuring that storage devices are recognized by the system and available to the user.

Symptom	Causes and Solutions
Missing devices	One of your first steps in troubleshooting is to verify the physical connectivity of devices. In the case of storage devices, ensure the data and power cables are connected. For external storage devices, verify that the USB, FireWire, or other data cables are connected, as well as the power cord. Linux will detect devices and dynamically create a device file in the <code>/dev</code> directory. You can check the <code>/dev</code> directory to verify whether Linux has detected the storage device.
Missing volumes	After confirming the physical device is connected, then verify the volumes created on the storage devices are recognized. Both the <code>parted</code> and the <code>fdisk</code> utilities can be used to verify that the partitions exist. In addition, the <code>/proc/partitions</code> file can be checked for basic information about the volumes. These tools can also be used to check whether a file system has been installed on the volume. The two common file systems will be ext4 or XFS.
Missing mount points	During the Linux startup process, the <code>/etc/fstab</code> file is checked to see what partitions should be automatically mounted. Mistakes in the <code>/etc/fstab</code> file will result in the volumes not being properly mounted or available to the users. The <code>mount</code> command can be used to manually attach storage volumes on the file system.

Symptom	Causes and Solutions
Degraded storage	Degraded storage refers to a situation where a storage drive in a RAID array has failed. Depending on the RAID type, the system may still be able to read from and write to the array, even with a failed drive. Typically, however, the overall performance of the system will be reduced by the failed drive. The array can be rebuilt with a replacement storage drive to return to optimal performance.
Performance issues	Investigating storage performance issues begins with considering the technology being used on the system. On workstations, SATA hard drives may be sufficient, though solid-state drives (SSDs) are also quite common. Servers typically require a much more efficient I/O system and a great deal more storage capacity. SCSI, SAS, and SSDs are a common choice for server storage due to their efficiency. These drives are usually part of a RAID array when deployed on servers. Even when a suitably high performance storage technology is chosen, a failing disk, controller, or other hardware component of the storage device can lead to performance issues.
Resource exhaustion	Each file that is opened in Linux is assigned a file descriptor that keeps track of the file when it is open. It is possible, on a very busy server where a great many files may be opened simultaneously, that the descriptors could all be consumed. One significant effect of this resource exhaustion is that most Linux commands will not run. You can resolve the issue by rebooting the server. The ulimit command can then be used to adjust the available number of file descriptors.
Storage integrity/bad blocks	Traditional magnetic hard disk drives may degrade over time. One common symptom of this aging is bad blocks, or sections of the hard disk drive that cannot be written to or read from. The file system will automatically mark these blocks and not attempt to use them to store data. If there are too many bad blocks, performance and storage capacity may be diminished. Too many bad blocks is also an indication of a failing hard disk drive. You should consider replacing the hard disk drive immediately.

THE ulimit COMMAND

The syntax of the **ulimit** command is **ulimit [options] [limit]**. For example, to set a limit for the maximum number of open file descriptors: **ulimit -n 512**

You can display all of the current limits by issuing **ulimit -a**

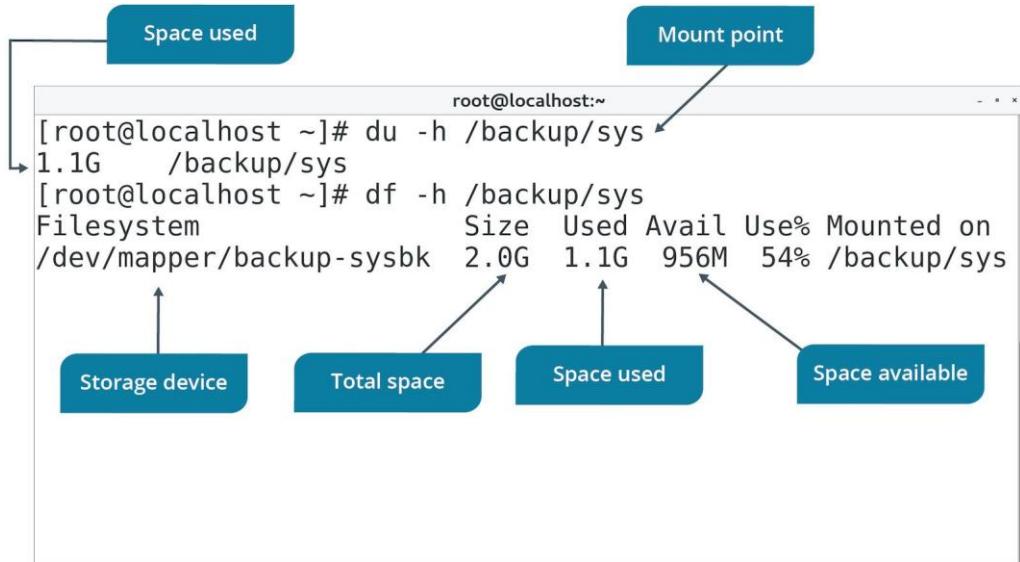
STORAGE SPACE TRACKING

The **df** and **du** commands facilitate storage space tracking. The **df** command ("disk free") enables you to view the device's free space, file system, total size, space used, percentage value of space used, and mount point. The **du** command ("disk usage") displays how a device is used, including the size of directory trees and files within it. It also enables you to track space hogs, which are directories and files that consume

large amounts of space on the storage drive. These are your go-to commands to confirm excessive storage space consumption that might be causing system issues.



Note: The `-h` option is useful because it makes the output more human-friendly (i.e., presenting size in units like GB rather than bytes).



Checking storage space on a device using both the `du` and `df` commands.

SYNTAX

The syntax of the `du` and `df` commands is `du/df [options] [object names]`

I/O SCHEDULING

I/O scheduling is the process by which the operating system determines the order of input and output operations as they pertain to block storage devices. Scheduling is important because, compared to CPU and memory operations, block storage operations are relatively slow—especially in disk-based technology like hard disk drives. The Linux kernel, therefore, doesn't just begin writing or reading to a drive in the order that such requests are submitted; instead, it prioritizes certain requests over others in order to minimize performance issues that can come with I/O tasks.

Although the kernel handles scheduling, you can actually configure the scheduler with different behavior types. Some behaviors are more appropriate than others in certain situations, and you may find that setting a new type increases read/write speeds.

Changing the scheduler is typically done during the troubleshooting process in order to finely tune storage performance when every bit of that performance matters.

SCHEDULER TYPES

The following table describes some of the different schedulers that are available to modern Linux kernel versions.

Type	Description
deadline	<p>This scheduler performs sorting of I/O operations using three queues: a standard pending request queue, a read first in first out (FIFO) queue, and a write FIFO queue; the latter two of which are sorted by submission time and have expiration values.</p> <p>When a request is submitted, it is sorted into the standard queue, and placed at the end of its appropriate FIFO queue. When the request at the top of the FIFO queue becomes older than the queue's expiration, the scheduler stops working with the standard queue and starts servicing requests from the top of the FIFO queue—in other words, it handles the oldest requests. This ensures that the scheduler doesn't "starve" a request for too long. This makes it ideal for certain workloads like multi-threaded workloads.</p>
cfq	<p>This refers to the Complete Fair Queuing (CFQ) scheduler. In this scheduler, each process is given its own queue and each queue has an interval by which it is accessed, or its time slice. The scheduler uses a round-robin system to access each queue and services requests from these queues until either their time slices or requests are exhausted.</p> <p>When this happens, the CFQ waits 10 milliseconds until it sees a new request on the queue, and if it doesn't, it moves on to another queue. Like the deadline scheduler, this helps to minimize request starvation. Its advantage is that it services processes fairly, and provides good performance in most workload situations. It is also the default scheduler for modern versions of the Linux kernel.</p>
noop	<p>This is the simplest scheduler and does not sort I/O requests, but merely merges them. This can be ideal in situations where the device or its storage controller performs its own sorting operations. It can also benefit devices that don't have mechanical components requiring seek time, like SSDs and USB flash drives, because this scheduler doesn't expend much effort in reducing seek time. However, the previous two schedulers are preferred in most other situations.</p>

SETTING THE SCHEDULER

You can set the scheduler to use on a particular device by modifying the `scheduler` file located at `/sys/block/<device name>/queue/scheduler`. Setting

the scheduler is as simple as echoing the desired option to this file, as in: `echo noop > /sys/block/sda/queue/scheduler`

Note that this sets the scheduler for runtime only; the setting will revert upon reboot. To persist your changes, you must modify the system's boot loader configuration.

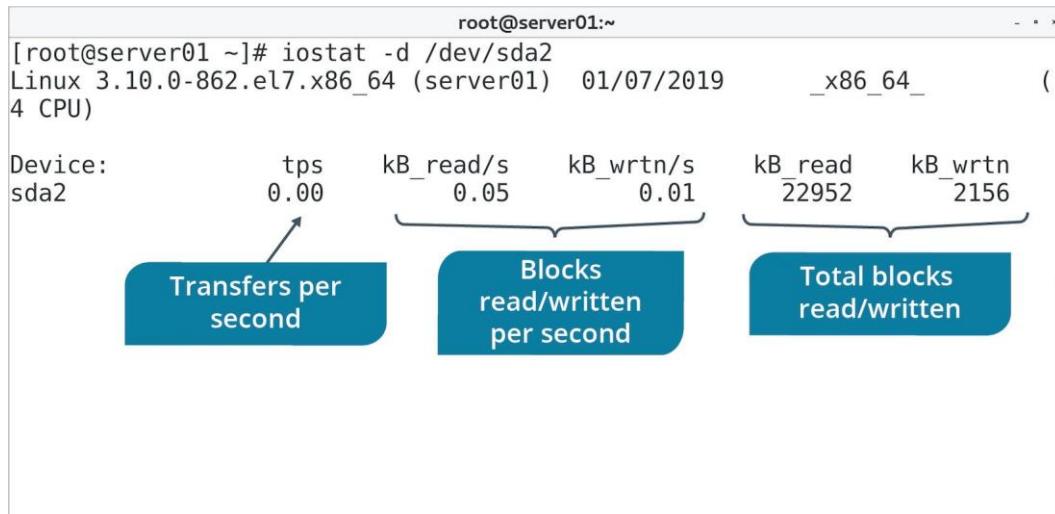
THE iostat COMMAND

The **iostat** utility generates reports on CPU and device usage. For storage, it provides input and output statistics for block devices and partitions. Using the **-d** option to specify device information only, the **iostat** command lists the following statistics for each storage device:

- Transfers (I/O requests) per second (**tps**).

- Number of blocks read per second (**kB_read/s**).
- Number of blocks written per second (**kB_wrtn/s**).
- The total number of blocks read (**kB_read**).
- The total number of blocks written (**kB_wrtn**).

You can use this report to monitor how a storage drive is being used and to identify any potential bottlenecks. For example, a faulty drive might have lower reads and/or writes per second than expected. You can also use the report to help you decide how to best distribute I/O load between the available devices.



Displaying storage device usage statistics.

SYNTAX

The syntax of the **iostat** command is **iostat [options] [device names]**

THE ioping COMMAND

The **ioping** command generates a report of device I/O latency in real-time. It will continuously "ping" the specified device with requests and print information about each request at the command-line. By default, this information tracks how long it took an I/O request to finish. Aside from specifying a device to test, you can also specify a path name to test whatever device is associated with that path.

Consider using **ioping** to troubleshoot latency issues with a storage devices, especially if you believe your read and/or write speeds are slower than they should be.



Note: This is similar to the standard **ping** command, which tests network latency.

```

root@server01 ~]# ioping -c 5 /dev/sda2
4 KiB <<< /dev/sda2 (block device 1 GiB): request=1 time=7.40 ms (warmup)
4 KiB <<< /dev/sda2 (block device 1 GiB): request=2 time=11.8 ms
4 KiB <<< /dev/sda2 (block device 1 GiB): request=3 time=12.5 ms
4 KiB <<< /dev/sda2 (block device 1 GiB): request=4 time=5.94 ms
4 KiB <<< /dev/sda2 (block device 1 GiB): request=5 time=9.51 ms

--- /dev/sda2 (block device 1 GiB) ioping statistics ---
4 requests completed in 39.8 ms, 16 KiB read, 100 iops, 402.5 KiB/s
generated 5 requests in 4.01 s, 20 KiB, 1 iops, 4.99 KiB/s
min/avg/max/mdev = 5.94 ms / 9.94 ms / 12.5 ms / 2.56 ms
[root@server01 ~]#

```

Total results

Pinging a storage device to test I/O latency.

SYNTA

X

The syntax of the **ioping** command is **ioping [options] {file/directory/device name}**

ioping COMMAND OPTIONS

The following table lists some options you can use with the **ioping** command.

Option	Used To
-c {count}	Specify the number of I/O requests to perform before stopping.
-i {time}	Set the time (interval) between I/O requests.
-t {time}	Set the minimum valid request time. Requests faster than this are ignored.
-T {time}	Set the maximum valid request time. Requests slower than this are ignored.
-s {size}	Set the size of requests.

STORAGE QUOTAS

A **storage quota** is the storage space that is allotted to a user for file storage on a computer. Storage quotas are configured on a per-user basis. File systems that implement storage quotas can have a soft limit, a grace period, and a hard limit. Once a user exceeds the soft limit, they are placed in the grace period for a default of seven days. The user is allowed to exceed this soft limit within this grace period, but cannot exceed the hard limit maximum. If the user goes below the soft limit, the timer resets. If the user still exceeds the soft limit when the timer expires, the soft limit is automatically imposed as a hard limit, and the user will be unable to use any additional storage.

Storage quotas are a good measure to prevent or respond to issues that arise due to excessive storage use. You can use these quotas to ensure that users are not consuming all of a drive's space and leaving none for other users or the system.

QUOTA MANAGEMENT COMMANDS

Quota management is the effective allotment and monitoring of quotas for all users. Linux has various commands that help ease the job of quota management for the system administrator.



Note: You should assign quotas in such a way that users are able to maximize the use of storage resources without data overflow.

Command	Used To
<code>quotacheck -cug {mount point}</code>	Create quota database files for a file system and check for user and group quotas.
<code>edquota -u {user name}</code>	Edit quotas for a specific user.
<code>edquota -g {group name}</code>	Edit quotas for a specific group.
<code>setquota -u {user name}</code>	Set quotas for a specific user.
<code>setquota -g {group name}</code>	Set quotas for a specific group.

QUOTA ACTIVATION

Before you can use these commands, you must actually activate user and/or group quotas on the file system. You can do this by editing the `fstab` file to add the options `usrquota` and `grpquota` to the relevant file system.

XFS QUOTAS

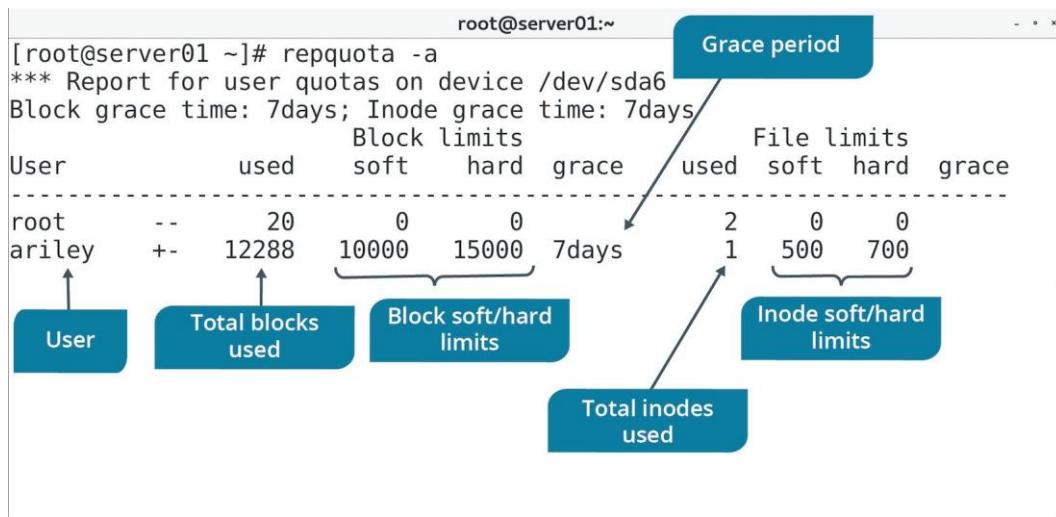
You can use the `xfs_admin` utility to configure quotas on XFS file systems. This utility can run in both interactive and non-interactive mode. When run non- interactively, use the `-C` option to specify which commands to run, and the `-X` option to enable expert mode, which is required for most administrative tasks. Tasks include setting limits on writing blocks and inodes, setting warning limits, generating quota reports, and more.

QUOTA REPORTS

Quota reports are created by the system so you can view storage space usage by each user. These reports enable you to check which user is taking up maximum disk space. They can also help you troubleshoot issues with quotas themselves—for example, quotas that are either too restrictive or too permissive.

A quota report contains the following details:

- The name of the user/group.
- The total number of blocks (in kilobytes) that are being used by the user/group on a file system.
- The user's/group's storage soft limit.
- The user's/group's storage hard limit.
- The grace period.
- The total number of inodes that have been used on a file system by the user/group.
- The soft limit on inodes.
- The hard limit on inodes.



A storage quota report.

REPORT GENERATION COMMANDS

Several commands are available for the generation of effective quota reports.

Command	Used To
repquota -a	Display the reports for all file systems indicated as read-write with quotas in the mtab file.
repquota -u {user name}	Display the quota report for a particular user.
quota -uv {user name}	Display the quota report for a particular user with verbose output.
warnquota -u	Check if users are not exceeding the allotted quota limit.
warnquota -g	Check if groups are not exceeding the allotted quota limit.

ADDITIONAL STORAGE TROUBLESHOOTING TECHNIQUES

Troubleshooting storage issues should start with the simple and work toward the more complex. Here are a few examples:

If a user claims they cannot create a file, verify that they have the appropriate permissions for the directory. From there, you might check to ensure the storage area is available (mounted partitions, both local and network), and that there is free space on the destination storage location. After those more simple steps, verify that the inode pool has not been exhausted by using the **df -i** command. If the inode pool has been exhausted, you'll need investigate the affected file system to see if it contains many unnecessary files, like temporary files, and delete them with the **rm** command.

In the event that a storage location appears to be unavailable, start your troubleshooting by verifying the physical connection for the storage device. From there, you would verify whether the storage device is recognized by the system—see the **/dev/** and **/proc/** directories for that information. You should also check configuration files for errors, including the **/etc/fstab** file. This is also true if you're mounting drives from network servers, such as NFS or Samba. Finally, you can consider using tools like **fsck**, the XFS toolset, or the ext4 toolset.

GUIDELINES FOR TROUBLESHOOTING STORAGE ISSUES

Use the following guidelines when troubleshooting storage issues.

TROUBLESHOOT STORAGE ISSUES

When troubleshooting storage issues:

- Ensure the devices are physically connected to the system.
- Ensure the devices are powered.
- Ensure the devices are turned on, if applicable.
- Verify the device is recognized by the system by checking the `/proc/` directory.
- Confirm that the configuration files do not contain any typographical errors.
- Ensure the configuration files have been reloaded if you have made changes to them.
- Confirm that there is enough storage capacity.
- Confirm that the I/O workload is not overwhelming the device.
- Use the `partprobe` command to cause the system to scan for new storage devices and partitions.

Activity 4-11

Discussing Storage Issues

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **What does the -h switch in the command df -h do for you?**
 2. **What does the iostat /dev/sda command do, and how can it be useful in troubleshooting?**
 3. **Which directory or file system is a likely candidate on which to impose storage quotas on a shared Linux system?**
 4. **As an administrator, you attempt to set quotas on /opt/finance for the finance group, but you receive the error that the mount point /opt/finance is not found or has no quota enabled. You verify that /opt/finance exists. You're trying to enable quotas, so it makes no sense that the error is that the file system has no quotas enabled. What is the problem?**
 5. **Why would an administrator use the ioping utility?**

Activity 4-12

Tracking Storage Space Usage

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created the /backup/data volume and filesystem.

SCENARIO

Lately, users have been complaining about poor performance and latency when they read files or write files to storage. Some users have also run into errors that say the storage device is full when they attempt to write to it. You spoke to some of these users to get more information, and a few of them mentioned that they were trying to back up data from their home directories when they ran into these errors. Using this information, you'll attempt to diagnose the storage issues on the Linux system.

1. Identify storage space usage.
 - a) Enter `df -h`
 - b) Verify that all of the file systems on the computer are listed, along with their usage data.
 - c) Enter `df -h /backup/data` to filter the results by the data backup volume.
 - d) Observe the total size of the volume, the number of bytes used, and the percentage of the volume that is used.
2. **What command might you issue to track the storage usage of files on the data backup volume?**
3. Track the storage space of the data backup volume.
 - a) Enter the command you used to answer the previous question.
 - b) Note the size of each object in this directory.
4. Identify the read/write statistics for the data backup volume.
 - a) Enter `iostat -d /dev/backup/databk`
 - b) Observe the I/O statistics for this drive.
5. **How might you use these statistics to diagnose issues with a storage drive?**

6. What command might you issue to track I/O latency in real-time by sending 100 requests to the data backup volume?

7. Which of the following Linux I/O schedulers is optimal in situations where a storage device performs its own I/O sorting operations, or in non-mechanical devices like SSDs and USB flash drives?
 - cfq
 - anticipatory
 - deadline
 - noop

8. Configure the I/O scheduler on the storage device.
 - a) Enter **sudo bash -c "echo scheduler > /sys/block/sda/queue/scheduler"** where *scheduler* is the scheduler you chose from the previous question.



Note: The *bash -c* portion tells Bash to run the entire command in a new shell as *sudo*

- b) Enter **cat /sys/block/sda/queue/scheduler** and verify that your scheduler was set.
The scheduler enclosed in brackets is the active one.

Activity 4-13

Configuring Storage Quotas

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you created the /backup/data volume and filesystem.

SCENARIO

After identifying excessive storage consumption on the data backup volume, you realize that most of these files were created by the **ariley** user. You decide to limit storage space usage for **ariley** on the data backup volume according to the following details:

- Block soft limit=10000
- Block hard limit=15000
- Inode soft limit=500
- Inode hard limit=700

These limitations will help prevent the performance and space consumption issues that users were experiencing.

1. Enable quotas for the data backup volume.
 - a) Enter **sudo vim /etc/fstab**
You'll need to edit the **fstab** file to configure the appropriate volume for quotas.
 - b) Press **i** to enter insert mode and scroll to the line that maps the **/dev/backup/databk** volume.
 - c) Edit this line so that it reads as follows (the change is in bold):
/dev/backup/databk /backup/data ext4 defaults,usrquota 0 0
 - d) Press **Esc**, then enter **:WQ** to save and quit.
 - e) Enter **sudo mount -o remount /backup/data** to remount the file system.
2. Configure the user quotas for the data backup file system.
 - a) Enter **sudo quotacheck -cugm /backup/data**
This creates the necessary quota files for the file system.
 - b) Enter **sudo quotaon -a** to turn on quotas for the file system.
3. Limit the ability for **ariley** to create data on the backup file system.
 - a) Enter **sudo edquota -u ariley**
 - b) Edit the configuration file to specify the following limitations:
/dev/mapper/backup-databk 0 10000 15000 0 500 700
 - c) Save and quit the file.

4. Start writing to the file system as the **ariley** user.
 - a) Enter **sudo chmod 777 /backup/*** to give **ariley** the appropriate permissions.
 - b) Enter **SU - ariley** to switch to the user.
 - c) Enter the password.
 - d) Enter **dd if=/dev/zero of=/backup/data/myfile bs=1M count=5**

This writes a 5 MB dummy file name **myfile** to the data backup directory.

5. Generate a quota report.
 - a) Enter **exit** to return to your student account.
 - b) Enter **sudo repquota /backup/data**
 - c) Verify that the report indicates how many blocks and inodes **ariley** is using, as well as the user's hard and soft limits.

```
*** Report for user quotas on device /dev/mapper/backup-databk
Block grace time: 7days; Inode grace time: 7days
          Block limits           File limits
User        used   soft    hard grace     used   soft    hard grace
-----
root       --     20      0      0          2      0      0
ariley     --      0  10000  15000        1  500  700
```

6. Exceed the soft limit quota.
 - a) Switch back to the **ariley** account.
 - b) Enter **dd if=/dev/zero of=/backup/data/myfile2 bs=1M count=7**
 - c) Verify that the results indicate that the user block quota has been exceeded.
7. **The user has reached the soft limit for storage blocks. What does this mean as far as the user's ability to write data to this file system?**

8. Exceed the hard limit quota.
 - a) Enter **dd if=/dev/zero of=/backup/data/myfile3 bs=1M count=10**
 - b) Verify that you were unable to write all of the data to the file system because you exceeded the hard limit quota.
9. Log out as **ariley** and return to your student account.

Summary

In this lesson, you managed your system's storage capabilities by creating partitions and logical volumes, as well as using various tools to modify storage devices and navigate the Linux directory structure. You also engaged in troubleshooting storage-based problems. With these skills, you can ensure your data is easy to access for those who need to access it, as well as easy to maintain and fix should issues arise.

Do you or would you prefer to work with standard partitions or logical volumes? Why?

What file system types have you worked with? What advantages and disadvantages have you encountered with each?



Practice Question: Additional practice questions are available on the course website.

Lesson 5

Managing Files and Directories

LESSON TIME: 2 HOURS, 30 MINUTES

LESSON INTRODUCTION

In the previous lesson, you created a foundation for your Linux® system by managing how the system and its data is stored. Now, you'll take a more high-level approach to managing the data itself. By managing files and directories, you'll be able to quickly create, retrieve, and otherwise process data so that it provides the most benefit to your organization and your day-to-day administrative tasks.

LESSON OBJECTIVES

In this lesson, you will:

- Create and edit text files.
- Search for files.
- Perform various operations on files and directories, including viewing, copying, and removing them.
- Process text files for easier analysis and use.
- Manipulate file output through redirection, piping, etc.

Topic A

Create and Edit Text Files



EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

Although you did some basic file creation and editing tasks earlier, it's time for you to become more familiar with text-based tools like Vim and GNU nano. Being able to create and edit files is a critical skill in Linux, as so much of what you configure on the system is done through text files.

TEXT EDITORS

A **text editor** is an application that enables you to view, create, or modify the contents of text files. Text editors were originally created to write programs in source code, but are now used to edit a wide variety of text-based files. Various types of text editors are compatible with Linux. However, text editors do not always support the formatting options that word processors provide. Text editors may work either in the CLI or GUI, and may have different modes of operation.

Text editors are important because, in Linux, most configuration components are text files: system configuration, network configuration, kernel configuration, shell environment configuration, etc. In the CLI, you'll be configuring most of these components by opening the relevant files in a text editor and adjusting some lines of text. So, being comfortable with a text editor is essential.

COMMON EDITORS

Many text editors are compatible with Linux. The following table lists some of the most common ones.

Text	Description
vi	A visual text editor that was originally created for Unix®, and was later cloned into FOSS versions.
Vim	The default text editor in most Linux distributions.
Emacs	A flexible, powerful, and popular text editor used in Linux and Unix.
gVim	The graphical version of the Vim editor.
gedit	A simple yet powerful GUI-based text editor used in the GNOME desktop environment.
GNU nano	A small, user-friendly text editor.

Vim

Vim, a contraction of Vi IMproved, is an extended version of the vi editor. Vim implements a text-based user interface to advanced text editing, and is favored by many system administrators and software engineers for its efficiency and ability to be extensively customized. Vim also includes useful features such as text completion, syntax highlighting, spell checking, and many more.

```

root@server01:~ root@server01:~ 
help.txt      For Vim version 7.4. Last change: 2012 Dec 06

VIM - main help file

Move around: Use the cursor keys, or "h" to go left,           k
              "j" to go down, "k" to go up, "l" to go right.   l
Close this window: Use ":q<Enter>".
Get out of Vim:   Use ":qa!<Enter>" (careful, all changes are lost!).

Jump to a subject: Position the cursor on a tag (e.g. bars) and hit CTRL-].
With the mouse:   ":set mouse=a" to enable the mouse (in xterm or GUI).
                  Double-click the left mouse button on a tag, e.g. bars.
Jump back:        Type CTRL-T or CTRL-O (repeat to go further back).

Get specific help: It is possible to go directly to whatever you want help
on, by giving an argument to the :help command.

help.txt [Help][RO]          1,1          Top
newfile                      0,0-1          All
"help.txt" [readonly] 221L, 8249C

```

The Vim interface.

THE vim COMMAND

The **vim** command invokes the Vim editor. However, the **vi** command may also be used for this purpose because it automatically redirects the user to Vim. When entered without a file name as an argument, the **vim** command opens a welcome screen by default. Use the syntax **vim {file name}** to open a file. If the file does not exist, Vim creates a file by the name specified and opens the file for editing. Vim supports multiple files being opened simultaneously.

WORKING WITH MULTIPLE WINDOWS

You can choose to display multiple files horizontally or vertically. Press **Ctrl+W+V** to create a vertical split, or press **Ctrl+W+S** to split the screen horizontally.

Vim MODES

Vim is a modal editor, and its different modes decide the functionality of various keys.

Mode	Enables Users To
Insert	Insert text by typing.
Execute	Execute commands within the editor.
Command	Perform different editing actions using single keystrokes.
Visual	Highlight or select text for copying, deleting, and so on.

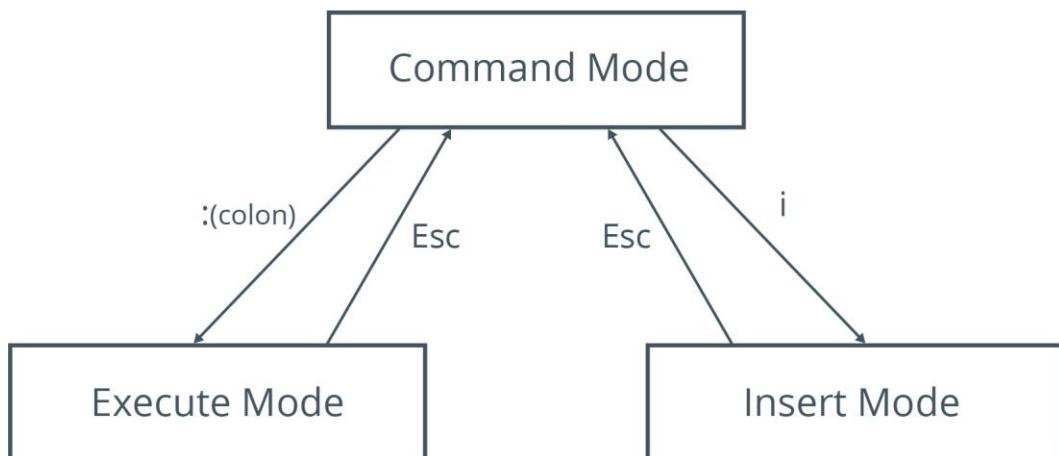
SWITCH MODES

Command mode is the default mode of Vim, but you can switch from command mode to any other mode by using a single keystroke.

Some of the keys to switch modes are listed here.

Key	Function
i	Switches to insert mode and inserts text to the left of the cursor.
A	Switches to insert mode and adds text at the end of a line.
I	Switches to insert mode and inserts text at the beginning of a line.

Key	Function
o	Switches to insert mode and inserts text on a new line below the cursor.
O	Switches to insert mode and inserts text on a new line above the cursor.
v	Switches to visual mode to enable selection, one character at a time.
V	Switches to visual mode to enable selection, one line at a time.
:	Switches to execute mode to enable users to enter commands.
Esc	Returns to command mode.



Switching between modes in Vim.

EXECUTE MODE COMMANDS

In command mode, when you enter the colon (**:**) operator, a small command prompt section appears at the bottom-left of the editor. This indicates that you are in execute mode and can run commands supported by Vim.

Some commands supported by Vim are listed in the following table.

Command	Function
:w {file} name}	Saves a file with a file name if it is being saved for the first time.
:q	Quits when no changes have been made after the last save.
:q!	Quits, ignoring the changes made.
:qa	Quits multiple files.
:wq	Saves the current file and quits.
:e!	Reverts to the last saved format without closing the file.
:!{any Linux command}	Executes the command and displays the result in the Vim interface.
:help	Opens Vim's built-in help documentation.

MOTIONS

Motions are single-key shortcuts that are used to navigate through files in command mode. These keys position the cursor anywhere within a document. They can be used for moving the cursor through characters, words, lines, or even huge blocks of text.

Navigation	Used To
h	Move left one character.
j	Move down one line.
k	Move up one line.
l	Move right one character.
^	Move to the beginning of the current line.
\$	Move to the end of the current line.
w	Move to the next word.
b	Move to the previous word.
e	Move to the end of the current word or to the end of the next word if you are already at the end of the word.
Shift+L	Move the cursor to the bottom of the screen.
Shift+H	Move the cursor to the first line of the screen.
(Line number) Shift+G	Move the cursor to the specified line number. gg
	Move the cursor to the first line of the file.
Shift+G	Move the cursor to the last line of the file.

NAVIGATION USING THE ARROW KEYS

In addition to using the **h**, **j**, **k**, and **l** keys to navigate through the editor, you can also use the **Up**, **Down**, **Left**, and **Right Arrow** keys. The conventional navigation keys such as **Home**, **End**, **Page Up**, and **Page Down** also work in Vim.

EDITING OPERATORS

Editing operators in command mode are powerful tools that can be used to manipulate text with simple keystrokes. They can also be used in combination with motions to edit multiple characters.

Some of the frequently used editing operators are listed here.

Editing	Used To
x	Delete the character selected by the cursor.
d	Delete text.
dd	Delete the current line.
p	Paste text on the line directly below the cursor.
P	Paste text on the line directly above the cursor.
/text string	Search through the document for specific text.
?text string	Search backward through the document for specific text.
y	Copy text.
yy	Copy the line directly above the cursor.
c{range of lines}c	Begin a change in the specified range. u
	Undo the latest change.

Editing	Used To
U	Undo all changes in the current line.
ZZ	Write the file only if changes were made, then quit the Vim editor.

COUNTS

A count is a number that multiplies the effect of keystrokes in Vim. It can be used in combination with motions, operators, or both. When used with a motion, cursor movement is multiplied according to the count specified. When used with editing operators, the action gets repeated the number of times specified.

The syntax for using a count with an operator and a motion is **operator [count] {motion}**

GNU nano

GNU nano is a small, user-friendly text editor that evolved from the Pico text editor created for Unix-like systems. It was added to the GNU Project shortly after its initial release. While Vim is a powerful text editor, it is not the most user-friendly, as evidenced by its multiple modes, bare interface, and many keystroke commands—some of which are unintuitive. The nano editor, on the other hand, is more visually helpful in that it displays its command shortcuts at the bottom of every open file.

Likewise, nano has fewer commands than Vim, and most command keystrokes share **Ctrl** as a common prefix. It also does not have different modes that you need to switch between.

Despite these advantages, nano lacks many of the features that make Vim so powerful, like split screen, text completion, syntax coloring, and more.

```

root@server01:~
GNU nano 2.3.1          File: newfile          Modified
Main nano help text

The nano editor is designed to emulate the functionality and ease-of-use of the UW Pico text editor. There are four main sections of the editor. The top line shows the program version, the current filename being edited, and whether or not the file has been modified. Next is the main editor window showing the file being edited. The status line is the third line from the bottom and shows important messages. The bottom two lines show the most commonly used shortcuts in the editor.

The notation for shortcuts is as follows: Control-key sequences are notated with a caret (^) symbol and can be entered either by using the Control (Ctrl) key or pressing the Escape (Esc) key twice. Escape-key sequences are notated with the Meta (M-) symbol and can be entered using either the Esc, Alt, or Meta key depending on your keyboard setup. Also, pressing Esc twice and then typing a three-digit decimal number from 000 to 255 will enter the character with the corresponding value. The

^Y Prev Page      ^P Prev Line      ^X Exit
^V Next Page      ^N Next Line

```

The GNU nano interface.

THE nano COMMAND

The **nano** command invokes the GNU nano editor. Without any arguments, the command will open a new file for editing, and you can later save this file with a specific name. Use the syntax **nano {file name}** to open an existing file. If the file does not exist, nano creates a file by the name specified and opens the file for editing. Like Vim, nano supports multiple files being opened simultaneously. These files are opened into different "buffers" that you can switch between.

nano SHORTCUTS

In GNU nano, the functions you use to work with text files and the editor itself are referred to as shortcuts. You activate most shortcuts by pressing the **Ctrl** key (represented as ^ in the editor) and then pressing the key that corresponds to the function you're trying to perform.

The following table lists some of the common nano shortcuts.

Shortcu	Used To
Ctrl+G	Open nano to the help screen.
Ctrl+X	Exit nano or close the current "buffer" (e.g., the help screen itself).
Ctrl+O	Save the currently open file.
Ctrl+J	Justify the current paragraph.
Ctrl+R	Insert another file into the current one.
Ctrl+W	Search the file.
Ctrl+K	Cut the currently selected line.
Ctrl+U	Paste the line that was cut.
Ctrl+C	Display the cursor's position.



Note: Despite the fact that the shortcuts are presented in uppercase, you shouldn't use the **Shift** key, only the **Ctrl** key.

NAVIGATION

Like other text editors, you can navigate in nano using the arrow keys, **Page Up**, **Page Down**, **Home**, etc. If you are missing these keys, nano also provides shortcuts for them, e.g., **Ctrl+V** to navigate to the next page and **Ctrl+Y** to navigate to the previous page.

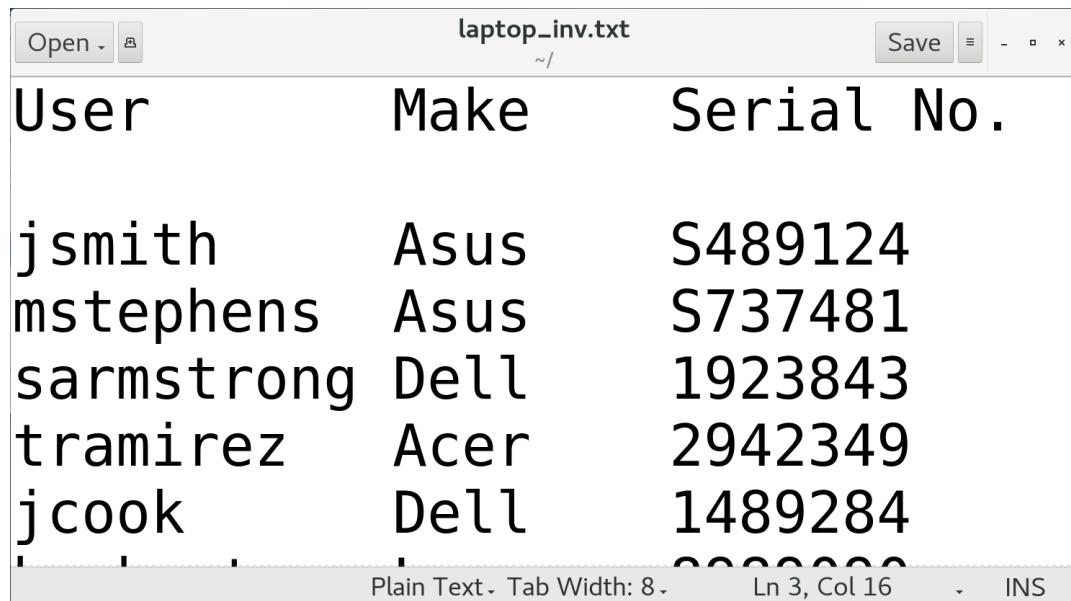
COPYING TEXT

Copying parts of text on a line requires you to "mark" the text you want to copy with the **Ctrl+^** shortcut. You then navigate your cursor to highlight the text you want to copy. Pressing **Alt+^** copies the marked/highlighted text, and **Ctrl+U** pastes it.

THE gedit TEXT EDITOR

The **gedit** text editor is the default text editor used in GNOME desktop environments and is a member of the GNU Project. Unlike Vim and nano, gedit has a GUI with a typical menu-based design that makes it easy to work with. It also has features like syntax highlighting and spell checking, and can be customized through plugins. While not as powerful as Vim, gedit may still be useful in systems that have a desktop environment installed.

Although you can launch gedit from the desktop, you can also use the CLI with the **gedit** command. The syntax is similar to **vim** and **nano**—no argument opens a new file, whereas providing a file name as an argument either opens an existing file or creates a new one with that name.



The screenshot shows the gedit text editor interface. The title bar reads "laptop_inv.txt" and "Plain Text". The window contains a table with three columns: "User", "Make", and "Serial No.". The data is as follows:

User	Make	Serial No.
jsmith	Asus	S489124
mstephens	Asus	S737481
sarmstrong	Dell	1923843
tramirez	Acer	2942349
jcook	Dell	1489284

At the bottom of the editor window, there are status indicators: "Plain Text", "Tab Width: 8", "Ln 3, Col 16", and "INS".

The gedit interface.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 5-1

Discussing Creating and Editing Text Files

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You're editing a file in Vim and you realize that you've made several errors and decide to start over with your editing. How do you leave the file without saving any changes that you've made?

2. You want to create a shell script using Vim. You enter Vim but can't type any letters. What do you need to do before you can begin typing?

3. You've created a script in Vim and note that you haven't entered any instructions or documentation into the file. Standard practice is to add your comments above the line of code that you're describing. How do you open a new line above the one you're currently typing on?

4. How do you copy and paste text in the Vim editor?

5. For text file editing in Linux, you have options other than Vim. Name one graphical editor and one other command-line interface (CLI) editor that you might use instead of Vim.

Activity 5-2

Creating Text Files

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

As one of the Linux server administrators, you've been asked to start a list of software that is installed or should be installed on the system. So, you'll create a text file and begin entering the names of software packages into it. You'll then save your work and pick up with the file later.

-
1. Create the software list file.
 - a) Enter `cd ~` to return to your home directory.
 - b) Enter `vim software_list.txt` to start editing a new file in Vim.
 2. Enter text into the file.
 - a) Press `i` to switch to insert mode.
 - b) Verify that the text "INSERT" is displayed at the bottom-left of the screen.
 - c) On the first line, enter **Apache HTTP Server**
 - d) Enter **MySQL** on the second line.
 - e) Enter **Eclipse** on the third line.
 - f) Type **OpenVAS** as the fourth and final entry.

Apache HTTP Server
MySQL
Eclipse
OpenVAS█

~
~
~
~
~
~

-- INSERT --

4 , 8

3. Save and view the text file.
 - a) Press **Esc** to return to command mode.
 - b) Enter **:WQ** to save the file and quit Vim.
 - c) Enter **cat software_list.txt** to view the file.

Activity 5-3

Editing Text Files

DATA FILE

/opt/linuxplus/managing_files_and_directories/software_list.txt

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

A colleague has taken your initial software list file and started filling it out. After he's done, you look it over to see if there are any mistakes that need correcting. You'll edit this file in both Vim and GNU nano to become more familiar with both text editors.

1. Copy the latest version of the software list.

- a) Enter the following:

```
cp -a /opt/linuxplus/managing_files_and_directories/
software_list.txt ~/
```

Note: Remember, you can use tab completion to speed up the process.



- b) Enter **ls** and verify that **software_list.txt** is listed.

2. Open the file in Vim and fix a spelling error.

- a) Enter **vim software_list.txt**

- b) Use the arrow keys to move the cursor down to the first instance of the text "Friefox".
- c) Position the cursor under the "i" in "Friefox".
- d) Press **x** to cut the letter "i".
- e) Move the cursor under the letter "F" and press **p** to paste the cut letter.
- f) Verify that the line now correctly says "Firefox".

3. Use the search functionality to find and fix the other instance of the spelling error.

- a) Enter **/Frie** to search for the next occurrence of the misspelled name.
- b) Correct the name so that it says "Firefox".

4. Fix the casing of one of the software names.

- a) Press **k** to go up line-by-line until you reach the line that says "openVAS" (note the lowercase "o").
- b) Press **^ (Shift+6)** to go to the beginning of the line.
- c) Press **x** to delete the first letter.
- d) Press **i** to enter insert mode, then type a capital **O**
- e) Press **Esc** to exit insert mode.

5. Delete a duplicate line and save the file.
 - a) Press **j** to go down line-by-line until you reach the second line that mentions "Apache".
 - b) Press **d** twice to delete the entire line.
 - c) Enter **:WQ** to save and close the file.
6. Open the file in GNU nano and make a correction.
 - a) Enter **nano software_list.txt**
 - b) Use the arrow keys to move to the "Y" under the "Configured?" column for "Eclipse".
 - c) Press **Delete**.
 - d) Type **N**
7. Remove a duplicate line.
 - a) Navigate down to the second instance of "LibreOffice".
 - b) Press **Ctrl+K** to cut the duplicate line.
8. Add another entry to the file.
 - a) Navigate to the beginning of a new line at the bottom of the file.
 - b) Type **Apache-Tomcat**
 - c) Press **Tab** until the cursor is under the "Version" column.
 -  **Note:** You can also use the spacebar for more precise alignment.
 - d) Type **9.0.12**
 - e) Place the cursor under the "Installed?" column and type **N**
 - f) Type **N** under the "Configured?" column.
9. Save the file and close GNU nano.
 - a) Press **Ctrl+O**.
 - b) Press **Enter** to save the file.
 - c) Press **Ctrl+X** to exit GNU nano.
 - d) Enter **cat software_list.txt** to view the file.

Topic B

Search for Files



EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

There will be times that you create a file and forget where in the directory structure you put it. There will also be times when you don't know the exact location of files created by the system, applications, or other users. In Linux, you have several powerful tools for finding the files you're looking for.

THE locate COMMAND

The **locate** command performs a quick search for any specified string in file names and paths stored in the **mlocate** database. This database must be updated regularly for the search to be effective. The results displayed may be restricted to files that users have permissions to access or execute.

```
student01@server01:~$ locate backup
/backup
/backup/data
/backup/log
/backup/sys
/backup/log/2019
/backup/log/auth
/backup/log/lost+found
/backup/sys/etc
/backup/sys/test
/backup/sys/test2
/backup/svs/etc/.pwd.lock
```

Searching for files and directories using the locate command.

SYNTAX

The syntax of the **locate** command is **locate [options] {string}**

locate COMMAND OPTIONS

The **locate** command supports different options that enable you to make your search more effective. Some of the options are described in the table.

Option	Used To
-r	Search for file names using regular expressions.
-C	Display only the number of matching entries found, rather than the file names.

Option	Used To
-e	Return only files that exist at the time of search.
-i	Ignore the casing in file names or paths.
-n {number of entries}	Return only the first few matches up to the specified number.

THE updatedb COMMAND

The **updatedb** command is used to build a database of files based on the **/etc/updatedb.conf** file. This command is used to update the **/var/lib/mlocate/mlocate.db** database. The **/etc/updatedb.conf** file consists of the paths that should be excluded while building the database. To add a path that needs to be excluded while building the database, open the **/etc/updatedb.conf** file and, in the **PRUNEPATH** variable, specify the path that need not be included while building the database. For example, **PRUNEPATH="/etc"** will exclude the **/etc** directory while building the database.

Though this is the default database searched by the **locate** command, there may be more databases containing file paths. If the database is not updated before performing a search, all files created after the last update will be excluded from the search.

THE find COMMAND

The **find** command enables you to search a specific location for files and directories that adhere to some search criteria. It recursively searches the directory structure, including any subdirectories and their contents, beginning with the search location you enter. You can perform one or more actions on the files found.

The **-type** option enables you to specify the type of object you're looking for, such as **d** for directory or **f** for file. The **-name** option is where you specify the name of the object you're looking for. The following example searches a user's home directory (and all subdirectories) for all files named **2019_report**:

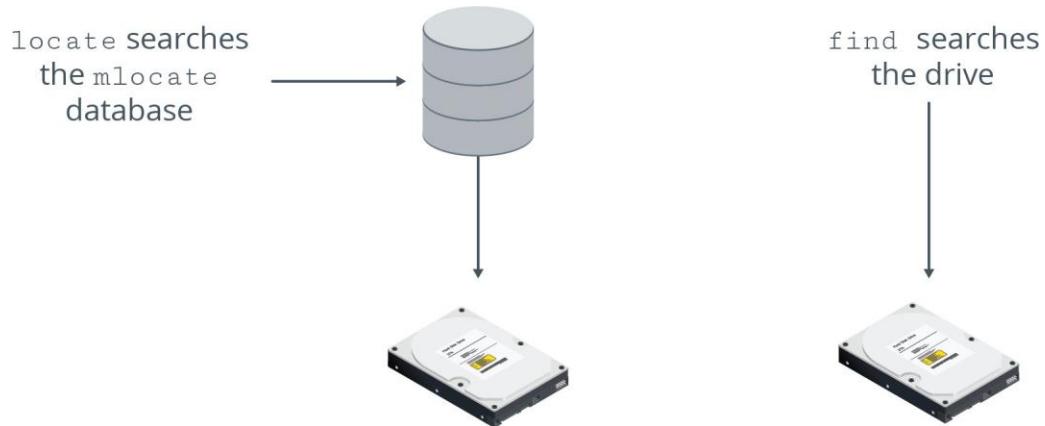
```
find /home/user -type f -name 2019_report
```

SYNTAX

The syntax of the **find** command is **find [options] {search locations} {search criteria} [actions]**

find VS. locate COMMANDS

The **locate** command searches a database and retrieves information on files present on your system. However, failure to keep this database updated may produce outdated results. The **find** command, on the other hand, performs a live search of the file system and may concentrate on a specific location. The **find** command may take more time to complete a search than the **locate** command.

*The find vs. locate commands.*

OPTIONS FOR FILES FOUND

When the system finds a listing that meets your criteria, there are several actions you can perform on the results. Several of these options are outlined in the following table.

Option	Action
-print	Displays the location of the files found.
-exec	Executes the command that follows.
-ok	Executes the command that follows interactively.
-delete	Deletes files found.
-fprint	Stores results in the target file.

THE which COMMAND

The **which** command displays the complete path of a specified command by searching the directories assigned to the **PATH variable**. For example, upon entering **which cat**, the following output is displayed: **/bin/cat**

```
student01@server01:~$ which cat
/bin/cat
```

Displaying the complete path of a command.

The **which** command can therefore help you locate where a program has been installed in case you need to modify this. It can also help you identify which version of a command you're using if there are multiple binaries of the command stored in different locations, one of which may be more ideal. By identifying where a command is running from, you can troubleshoot unexpected behavior from that command.

SYNTAX

The syntax of the **which** command is **which [options] {program names}**

THE **whereis** COMMAND

The **whereis** command is used to display various details associated with a command. For example, when entering **whereis ls** the following output is displayed: **ls: /bin/ls /usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz**

Where **/bin/ls** indicates the location of the **ls** command and **/usr/share/man/man1/ls.1.gz /usr/share/man/man1p/ls.1p.gz** indicates the location of the man pages for the **ls** command.

```
root@server01:~#
[root@server01 ~]# whereis ls
ls: /usr/bin/ls /usr/share/man/man1/ls.1.
gz /usr/share/man/man1p/ls.1p.gz
```

Displaying location information for a command.

SYNTA

X

The syntax of the **whereis** command is **whereis [options] [directory name] {file name}**

whereis COMMAND OPTIONS

The **whereis** command has several options, as described in the following table.

Option	Used To
-b	Search only for binaries.
-m	Search only for manual sections.
-s	Search only for sources.
-u	Search for unusual entries.



Note: To learn more, check the **Video** tab on the course website or any videos that supplement the content for this lesson.

Activity 5-4

Discussing Searching for Files

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You tried using `locate` to find README files on your new Linux system but quickly realized that it wasn't installed at build time. After installation, you tried using `locate` to find README files, but again the command has failed to find anything. You discover that for `locate` to work, you must build the `mlocate` database. Which command can you run to build the `mlocate` database?
2. You tried using `locate` to find the `cat` command but `locate` searches for strings and returned more than 2,000 hits. You need something more efficient for searching for a specific file, especially commands. Which command can you run to selectively find the `cat` command?
3. You and other junior Linux administrators have noticed that some basic commands don't have man pages associated with them. You would like to know which ones do. One of the other administrators told you about a command that would also list the man pages for a command search. To which command was he referring?
4. The `find` command is one of the most powerful in a system administrator's toolbox. What does the following `find` command do? `find / -name "carbon*" -print`
5. Which of the search commands at your disposal would you use to locate a user's lost files, knowing that the user has access to dozens of directories and subdirectories as a member of several organizational groups?

Activity 5-5

Searching for Files

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

One of your duties as a Linux administrator is to ensure your system logs are functioning as expected. These logs are crucial to diagnosing issues and identifying other unwanted behavior. So, to start, you'll search for where the log files are stored on your system. Then, you'll begin to search for logs that meet specific size requirements and have been recently updated. That way you'll be able to confirm which logs are continuously recording a significant amount of information, as expected.

1. Search for the location of system log files.
 - a) Enter **sudo find / -type d -name 'log'**
 - b) Verify that there are several locations on the root volume that contain the word "log".
 - c) Enter **sudo find / -type f -name 'messages'**
 - d) Verify that the location of the messages log is identified as **/var/log/messages**

2. Search for log files that are above 100 KB in size.
 - a) Enter **sudo find /var/log -type f -size +100k**
 - b) Note one of the files in the results and enter **ls -lh /var/log/<file name>** to verify that it is indeed above 100 KB in size.

3. Search for log files that have been updated within the last 30 minutes.
 - a) Enter **sudo find /var/log -type f -mmin -30**
 - b) Verify that one of the files in the results has a timestamp within the last 30 minutes.



Note: You can use **ls -Z** on the file to do this.

4. Search for log files that are either empty or above 100 KB, *and* have been updated in the last 30 minutes.
 - a) Enter **sudo find /var/log -type f -size 0 -or -size +100k -mmin -30**
 - b) Verify that these conditions are accurate for at least one of the files.

5. **What are some advantages of using the `find` command over using the `locate` command?**
-

Topic C

Perform Operations on Files and Directories



EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.

There are many ways you can manage a file once you've created and edited it to your liking. Using various tools, you'll ensure that your files are in the state you expect them to be in.

THE cat COMMAND

The **cat** command, short for concatenate, can display, combine, and create text files. It is most frequently used to display the contents of small text files, as it does not have a screen scrolling capability.

Some of the **cat** command options are described in the following table.

Option	Used To
-n	Precede the output with its respective line number.
-b	Number the lines, excluding the blanklines.
-s	Suppress output of repeated empty lines.
-v	Display non-printing characters as visible characters, other than tabs, new lines, and form feeds.
-e	Print a \$ character at the end of each line, prior to the new line.
-t	Print tabs as ^I and form feeds as ^L

SYNTAX

The syntax of the **cat** command is **cat [options] {file names}**

THE head AND tail COMMANDS

The **head** command displays the first 10 lines of each file. The **tail** command displays the last 10 lines of each file. These commands are useful when you only need to see the beginning or the end of a file. For example, you can check recent log entries by viewing the last 10 lines of a log file.

10 most recent log entries

```

root@server01:~# tail /var/log/secure
Jan  7 18:49:27 server01 su: pam_unix(su-l:session): session opened for user ariley by
student01(uid=0)
Jan  7 18:49:38 server01 su: pam_unix(su-l:session): session closed for user ariley
Jan  7 18:58:32 server01 gdm-password: gkr-pam: unlocked login keyring
Jan  7 19:06:39 server01 su: pam_unix(su-l:session): session opened for user student01
by student01(uid=0)
Jan  7 19:06:42 server01 su: pam_unix(su-l:session): session closed for user student01
Jan  7 19:07:58 server01 su: pam_unix(su-l:session): session closed for user root
Jan  7 19:07:59 server01 su: pam_unix(su-l:session): session closed for user manderson
Jan  7 19:08:00 server01 su: pam_unix(su-l:session): session closed for user root
Jan  7 19:08:01 server01 su: pam_unix(su-l:session): session closed for user manderson
Jan  7 19:33:24 server01 su: pam_unix(su-l:session): session opened for user root by s
tudent01(uid=1000)
[root@server01 ~]#

```

Displaying the last 10 lines of a file.

SYNTA

X

The syntax of the **head** and **tail** commands is **head/tail [options] {file names}**

tail COMMAND OPTIONS

The following are some common options used with the **tail** command:

- **-f**—dynamically watch a file (the output will automatically update when the file changes).
- **-n {number}**—show the specified number of lines, rather than the default of 10. Can also be used with the **head** command.

THE less AND more COMMANDS

Both the **less** and **more** commands are similar in that they enable you to display the contents of a file and page through those contents if they extend beyond the screen. The **less** command typically has additional features that **more** doesn't, but newer versions of **more** have added some of those features. While you're free to use either command, the **less** command is generally preferred.

First page

```

root@server01:~ [ESC][32m OK [ESC][0m Started Show Plymouth Boot Screen.
[ESC][32m OK [ESC][0m Reached target Paths.
[ESC][32m OK [ESC][0m Reached target Basic System.
[ESC][32m OK [ESC][0m Found device /dev/mapper/centos-root.
Starting File System Check on /dev/mapper/centos-root...
[ESC][32m OK [ESC][0m Started File System Check on /dev/mapper/centos-root.
[ESC][32m OK [ESC][0m Started dracut initqueue hook.
[ESC][32m OK [ESC][0m Reached target Remote File Systems (Pre).
[ESC][32m OK [ESC][0m Reached target Remote File Systems.
Mounting /sysroot...
[ESC][32m OK [ESC][0m Mounted /sysroot.
[ESC][32m OK [ESC][0m Reached target Initrd Root File System.
Starting Reload Configuration from the Real Root...
[ESC][32m OK [ESC][0m Started Reload Configuration from the Real Root.
[ESC][32m OK [ESC][0m Reached target Initrd File Systems.
[ESC][32m OK [ESC][0m Reached target Initrd Default Target.
Starting dracut pre-pivot and cleanup hook...
[ESC][32m OK [ESC][0m Started dracut pre-pivot and cleanup hook.
Starting Cleaning Up and [ESC][32m OK [ESC][0m cleanup hook.
/var/log/boot.log-20190102 ← Indicates more pages

```

Paging through a long file.

SYNTAX

The syntax of the **less** and **more** commands is **less/more [options] {file names}**

less COMMAND OPTIONS

The following table lists some of the options for the **less** command.

Option	Used To
-e	Exit the program the second time it reaches the end of the file.
-E	Exit the program the first time it reaches the end of the file.
-i	Ignore case in searches.
-n	Suppress line numbers.

NAVIGATION

Navigation in **less** uses many of the same commands you've seen before, like the arrow keys to scroll line-by-line and **Page Up** and **Page Down** to scroll by page. You can also use **/** to search a file for a particular text string, and press **n** and **N** to move to the next or previous instance of the searched string. Press **q** to quit the program.

THE cp COMMAND

The **cp** command enables you to copy and then paste a file or directory. The initial object is left where it is, but an exact duplicate of that object is created at the destination you specify. When you copy directories, you must specify the **-R** option to copy the specified directory recursively.

SYNTAX

The syntax of the **cp** command is **cp [options] {file/directory name to copy} {file/directory name destination}**

For example, to copy the `~/myfile`s directory and its contents to `/opt/myfile`s:

```
cp -R ~/myfile /opt/myfile
```

THE mv COMMAND

The `mv` command moves files and directories to other locations. It is similar to the `cp` command, but does not leave the initial object in place. Therefore, `mv` is more like a cut and paste operation.

The Bash shell does not have a dedicated rename command, but instead uses `mv` to accomplish that function. The act of "moving" a file or directory and supplying a new name as the destination essentially renames that object.

SYNTAX

The syntax of the `mv` command is `mv [options] {file/directory name to move} {file/directory name destination}`

For example, to move `~/file1` to `/opt/file1`:

```
mv ~/file1 /opt/mylist
```

For renaming purposes, the syntax is `mv [options] {old file/directory name} {new file/directory name}`

THE touch COMMAND

The `touch` command changes the time of access or modification time of a file to the current time, or to the time specified in an argument. It is also used to create an empty file with the specified file name, assuming the file does not exist. This is often useful in testing permissions or in simply creating files that will later be processed by some application.

SYNTAX

The syntax of the `touch` command is `touch {file names}`

THE rm COMMAND

The `rm` command removes files and directories. You must use the `-R` option to recursively remove files, subdirectories, and the parent directory itself.

SYNTAX

The syntax of the `rm` command is `rm [options] {file/directory names}`

For example, to remove the `~/myfile`s directory and its contents:

```
rm -R ~/myfile
```

THE unlink COMMAND

The `unlink` command is similar to the `rm` command, but can only remove one file at a time and cannot remove directories.

THE ls COMMAND

At this point, you've seen much of the power of the **ls** command to list the contents of directories, as well as the permissions information of directories and files. The following table summarizes some of the most useful options that are available with the **ls** command.

Option	Used To
-l	Display a long list including the permissions, number of hard links, owner, group, size, date, and file name.
-F	Display the nature of a file, such as * for an executable file and / for a directory.
-a	Display all files present in the directory, including the files whose names begin with a period (.).
-R	Recursively display all subdirectories.
-d	Display information about symbolic links or directories rather than the link's target or the contents of the directory.
-L	Display all files in a directory, including symbolic links.

SYNTAX

The syntax of the **ls** command is **ls [options] [file/directory names]**

ls COLORS

In the Bash shell, when you execute the **ls** command, you may have noticed that the results sometimes appear in different colors. These colors distinguish different types of files. By default, some of the colors are:

- Default color: Normal/text file
- Blue: Directory
- Sky blue: Symbolic link or audio file
- Green: Executable file
- Yellow with black background: Device
- Pink: Image file
- Red: Archive file
- Red with black background: Broken link

THE mkdir AND rmdir COMMANDS

The **mkdir** command is used to create (or make) a directory. You supply the name of the directory as an argument. The **rmdir** directory is used to remove directories, but only those that are empty (i.e., contain no files or subdirectories). In order to delete a directory with actual contents, you must use the **rm -R** command.

SYNTAX

The syntax of the **mkdir** and **rmdir** commands is **mkdir/rmdir {directory names}**



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 5-6

Discussing Performing Operations on Files and Directories

SCENARIO

Answer the following questions to check your understanding of the topic.

-
1. **Linux users use the `cat` command to print a file's contents to the screen, but the original purpose of `cat` is to concatenate or join the contents of two or more files together into a single file. You have files `test.1` and `test.2` that you want to combine into a single file, `test.3`. How can you accomplish this using `cat`?**

 2. **As a system administrator, what application of the `tail` command comes to mind first for checking on system status or an application's status?**

 3. **There are two utilities that are good for paging through very long files with search capability. Both enable you to stop, proceed line-by-line, page-by-page, and to page up. Some administrators prefer one utility over the other because it has a few more options. What are these two utilities?**

 4. **Your manager asks you to copy all the files in the `/opt` directory to `/backup` for further inspection and analysis. Rather than copying every file and subdirectory manually, how can you copy `/opt` and all its contents to `/backup` with a single command?**

 5. **Diane Smith has left the company and you want to preserve her home directory, so you decide to rename it. How can you rename `/home/dsmith` to `/home/dsmith.keep`?**
-

Activity 5-7

Reading Files

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

Another one of your duties is, naturally, to review the system's log files. But before you dive into log analysis, you need to determine the best way to display text files for reading. So, you'll use commands like **cat** and **less** to see where each one can come in handy.

1. Open the messages log for reading.
 - a) Enter **Sudo cat /var/log/messages**
 - b) Notice that the file's contents scroll past the screen many times, indicating that the file is too large to read from a single screen.
 - c) Enter **Sudo less /var/log/messages**
 - d) Verify that only the first page of the file printed to the screen, and at the bottom of the screen, the name of the file is highlighted.
 - e) Press the **Down Arrow** to scroll down a single line.
 - f) Press the **Up Arrow** to scroll back up a single line.
 - g) Press **Spacebar** to scroll down an entire page.
 - h) Press **Page Up** to scroll up an entire page.
2. Search for a specific string in the log.
 - a) Enter **/SELinux**
 - b) Verify that the file jumps to the first instance of this text string, and that it is highlighted on the top line.

```
Dec 11 13:30:39 localhost kernel: SELinux: Initializing.
Dec 11 13:30:39 localhost kernel: Yama: becoming mindful.
Dec 11 13:30:39 localhost kernel: Dentry cache hash table
  6 (order: 11, 8388608 bytes)
Dec 11 13:30:39 localhost kernel: Inode-cache hash table
  (order: 10, 4194304 bytes)
Dec 11 13:30:39 localhost kernel: Mount-cache hash table
  (order: 5, 131072 bytes)
Dec 11 13:30:39 localhost kernel: Mountpoint-cache hash t...
```

- c) Press **n** to view the next instance of this text string in the file.
- d) Press **N** (note the capitalization) to navigate to the previous instance of the search term in the file.
- e) Press **q** to quit reading the file.



Note: Remember, you can enter `clear` to clear the screen as needed.

3. Print only the first and last lines of the log.
 - a) Enter `sudo head /var/log/messages`
 - b) Verify that the first 10 lines of the log were printed to the screen.
 - c) Enter `sudo tail /var/log/messages`
 - d) Verify that the last 10 lines of the log were printed to the screen.
 4. **Why might printing only the first or last few lines be preferable to reading the entire file?**
-

Activity 5-8

Manipulating Files and Directories

DATA FILE

All files in:

/opt/linuxplus/managing_files_and_directories/aups

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You've been asked to move some corporate policy documents from the HR lead's workstation to a Linux server. The policies should be more centrally available and not dependent on one particular person's system. The HR lead admits that she didn't do a great job organizing the policy documents, as several older versions of acceptable use policies (AUPs) are mixed in with more current, active versions, and the old policies were written before she implemented a consistent naming convention. All of the documents are in a single directory named **aups**.

First, you'll need to copy the documents to your home directory as a temporary staging area. You'll then organize these policy documents by retaining only the most recent ones and deleting older ones that no longer apply. You've also been told that more types of policies will need to be located on the server, other than AUPs. So, you'll effectively rename the **aups** folder to the more general **policies** and create some placeholder files. Later, when you receive more policies to add, you'll be able to deploy the directory where other authorized users can reach it.

1. Copy the policy files.

- a) Enter the following:

```
cp -r /opt/linuxplus/managing_files_and_directories/aups ~/
```

This copies the **aups** directory to your home directory, leaving its original location intact.

- b) Enter **cd aups**

- c) Enter **ls -l** and verify there are five files, three of which are marked as "OLD" and have inconsistent file names.

```
[student01@localhost aups]$ ls -l
total 20
-rwxr-xr-x. 1 student01 student01 1472 Dec 14 14:15 aup_v1.txt
-rwxr-xr-x. 1 student01 student01 1878 Dec 14 14:15 aup_v2.txt
-rwxr-xr-x. 1 student01 student01  794 Dec 14 14:15 OLD acceptable_use2.txt
-rwxr-xr-x. 1 student01 student01  353 Dec 14 14:15 OLD acceptableuse.txt
-rwxr-xr-x. 1 student01 student01 1015 Dec 14 14:15 OLD acct use3.txt
```

2. Create a new directory and move the most recent policy files into it.

- a) Enter **mkdir ..//policies** to create a new directory.
 - b) Enter **mv aup_v1.txt ..//policies**
 - c) Enter **mv aup_v2.txt ..//policies**
 - d) Enter **ls -l** and verify that these two files are no longer in this directory.
3. Verify that the recent policies were moved to the new folder.
 - a) Enter **cd ..//policies**
 - b) Enter **ls -l** and verify that the two recent files are in this directory.
 4. Create placeholder files for future policies.
 - a) Enter **touch user_sec_policy.txt**
 - b) Enter **ls -l** and verify that a blank file with this name was created.
 - c) Use **touch** to create three more blank files in **~/policies** with the following names:
server_sec_policy.txt
email_policy.txt
clean_desk_policy.txt
 - d) Enter **ls -l** and verify that the files exist.
 5. Delete the **aups** folder and its contents as they're no longer needed.
 - a) Enter **rmdir ..//aups**
 - b) Verify that you cannot remove this object because it is a directory.
You need to specify the **-R** (recursive) option with **rm** in order to delete non-empty directories.
 - c) Enter **rm -R ..//aups**
 - d) Enter **ls ..** and verify that the **aups** directory is gone, as are the old policy files.
-

Topic D

Process Text Files



EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

Beyond performing basic file operations like reading, moving, and copying, you can also manipulate files so that they are more useful to you. In this topic, you'll process files so that they're easier to work with based on certain business needs.

THE echo COMMAND

The `echo` command is used to display a line of text on the terminal. You can also use the `echo` command to write text to a file by providing the string after the `echo` command and redirecting to the file.

```
root@server01:~#
[root@server01 ~]# echo 'Hello, World!'
Hello, World!
```

Echoing a string of text.

SYNTA

X

The syntax of the `echo` command is `echo {string}`

THE printf COMMAND

The `printf` command is similar to `echo`, but provides the user with much more control over how the output is formatted. You can supply various format characters within the text you want to output, using a backslash (\) to indicate when they are being used. For example: `printf "Hello.\nWhat's your name?"` will print:

Hello.

What's your name?

This is because \n is the newline format character, and automatically adds a new line wherever it is placed.

The **printf** command also supports conversion characters, which use a percent sign (%) to indicate when they are being used. Conversion characters are typically used in scripts to change the output of a variable, like dictating the number of decimal places to print after a precise calculation.

THE tr COMMAND

The **tr** command is used to translate a string of characters. It is predominantly used to change the case of letters in a file. This command acts only on a stream of characters and does not accept file names as arguments. You must use redirection to actually change a file.

SYNTAX

The syntax of the **tr** command is **tr {character 1} {character 2}** where {character 1} is the character to be replaced.

THE wc COMMAND

The word count (WC) command is used to count the number of lines, words, and characters in a text file. If multiple files are specified, then the command displays the counts for each file and the total count for all files.

```
student01@server01:~$ wc laptop_inv.txt
14 40 352 laptop_inv.txt
```

Counting characters in a file.

SYNTAX

The syntax of the **WC** command is **wc [options] {file names}**

wc COMMAND OPTIONS

The WC command provides various options that enable you to specify the nature of the output.

Option	Used To
-C	Display the byte count.
-m	Display the character count.

Option	Used To
-l	Display the newline count.
-w	Display the word count.

THE sort COMMAND

The **sort** command arranges the lines in a file. Common **sort** command options are provided in the table.

Option	Used To
-k{column numbers}	Specify field values. For example, -k2 indicates the second field.
-n	Compare and sort lines based on the string numerical value.
-r	Sort fields in descending order. By default, the fields are sorted in ascending order.
-t{delimiter}	Separate one field from another.



Note: A delimiter can be a tab, space, colon, semicolon, period, or comma used to separate one field from another.

```
student01@server01:~ [student01@server01 ~]$ sort -k2 laptop_inv.txt
tramirez    Acer      2942349
manderson   Acer      2988481
lbarnes     Asus      S393892
jsmith       Asus      S489124
mstephens   Asus      S737481
nporter     Asus      S892849
kriley      Dell      1390390
jcook        Dell      1489284
sarmstrong  Dell      1923843
tlee         Lenovo    8112091
```

Sorting a text file.

SYNTA X

The syntax of the **sort** command is **sort [options] {file names}**

THE cut COMMAND

The **cut** command extracts the specified lines of text from a file. Common **cut** command options and their uses are given in the following table.

Option	Used To
-c	Specify the number of the character to cut from each line.

Option	Used To
-d{delimiter}	Separate one field from another.
-f{field numbers}	Specify the field numbers to cut on as separated by the delimiter. For example, -f2 indicates the field between the first and second instances of the delimiter.
-s	Suppress a line if the delimiter is not found.

The screenshot shows a terminal window with the following command and output:

```
[root@server01 ~]# cut -d: -f1-3 /var/log/secure
Jan 7 13:28:57 server01 gdm-password]
```

Annotations explain the command parameters:

- Use colon as delimiter**: Points to the `-d:` option.
- Extract fields 1 through 3**: Points to the `-f1-3` option.
- Field 1**, **Field 2**, **Field 3**: Indicate the extracted fields from the log entry.

Below the terminal window, the extracted fields are shown:

```
Jan 7 13:38:56 server01 pkexec[17482]
Jan 7 13:39:25 server01 su
Jan 7 13:39:33 server01 su
Jan 7 13:44:02 server01 su
Jan 7 13:49:59 server01 su
Jan 7 13:56:06 server01 su
Jan 7 14:06:59 server01 gdm-password]
```

Extracting a portion of a log.

SYNTAX

The syntax of the **cut** command is **cut [options] {file names}**

THE paste COMMAND

The **paste** command is used to merge lines from text files horizontally. Each line of an initial file is a row in the first column; using **paste**, you specify a second file, and every line of the second file becomes a row in a new, second column. By default, the **paste** command uses a tab space delimiter to separate each column. You can use the **-d** option to specify a different delimiter.

For example, you have a file named **Cities**:

```
New York
Tokyo
London
Lima
```

You also have a second file named **Countries**:

```
United States
Japan
England
Peru
```

The output of **paste -d , cities countries** is as follows:

New York, United States
 Tokyo, Japan
 London, England
 Lima, Peru

THE **diff** COMMAND

The **diff** command is used to compare text files. The command displays the two files and the differences between them. Using various symbols, the output suggests how you can change one file to make it identical to the other. Each symbol has a special meaning.

The less than symbol (<) with a line after it means that line should be removed from the first file because it doesn't appear in the second. The greater than symbol (>) with a line after it means that line should be added from the second file. In addition, the **diff** command also denotes the line numbers for each file that would be affected by deletion, addition, and change operations.



Note: The *diff* command doesn't actually make any changes on its own.

Difference (by line number)

```
student01@server01:~/policies
[student01@server01 policies]$ diff aup_v1.txt aup_v2.txt
33a34,41
> Rule: BK-01 Rules Relating to Data Backup
>
> 1. Ensure you are you backing up all criti
eparate partition or storage drive.
> 2. Only use external storage hardware that is provided by
IT.
> 3. Ensure all critical data is encrypted in backups.
> 4. Perform data backups at a monthly frequency or less.
> 5. Disconnect backup drives when not in use.
>
35a44
> 2018-10-06      Mary Stephens Added data backup rules.
```

Add text for files to be identical

Files being compared

Comparing two text files.

SYNTA

X

The syntax of the **diff** command is **diff {file name 1} {file name 2}**

diff COMMAND OPTIONS

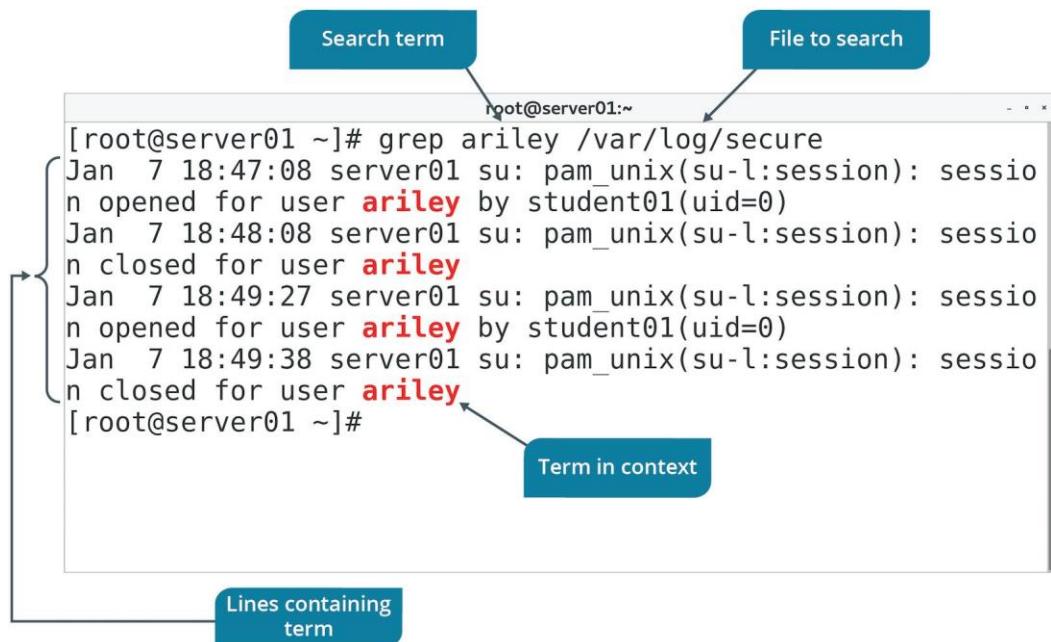
The **diff** command has various options that enable you to specify the nature of the output.

Option	Used To
-b	Ignore spacing differences.
-i	Ignore case differences.
-t	Expand tab characters in output lines.
-w	Ignore spacing differences and tabs.
-C	Display a list of differences with three lines of context.
-U	Output results in unified mode, which presents a more streamlined format.

THE grep COMMAND

The **grep** command, in its most basic form, is a search tool. Unlike **find** or **locate**, it is not limited to finding file names; it is most often used to search the contents of a file for a particular string of text. As output, **grep** displays each full line of the file that your search pattern was found in. In this way, you can use **grep** to both process a text file and read the contents that are most pertinent to you. For example, you may want to audit a user's login events by looking at an access log.

Instead of reading the entire log or stepping through a search term in a text editor, you can simply print all of the relevant lines to the screen with the **grep** command.



Searching a file for lines containing a specific term.

SYNTAX

The syntax of the **grep** command is **grep [options] {search pattern} {file names}**

grep COMMAND OPTIONS

The **grep** command has many options. Several common ones are described in the following table.

Option	Used To
-E {pattern}	Match a pattern as an extended regular expression (ERE).
-F {pattern}	Match a pattern as a list of fixed strings.
-f {file name}	Match patterns contained in the specified file.
-i	Ignore casing.
-v	Output only lines that <i>don't</i> match the provided pattern.
-c	Only print the number of matching lines, not the lines themselves.
-I	Only print the file(s) that have matching lines, not the lines themselves.
-o	Only print the matching part of a line, not the entire line.

USING grep TO FIND FILES

In addition to searching the contents of files, you can use grep to search a directory in order to locate a certain file. The **ls -l | grep audit** command returns a long listing of any files in the current directory whose name contains "audit".

THE egrep COMMAND

The **egrep** command is essentially the same as the **grep -E** command. However, **egrep** is deprecated, as **grep -E** is the preferred syntax.



Note: Regular expressions are beyond the scope of this course. For more information, navigate to <https://www.regular-expressions.info/quickstart.html>.

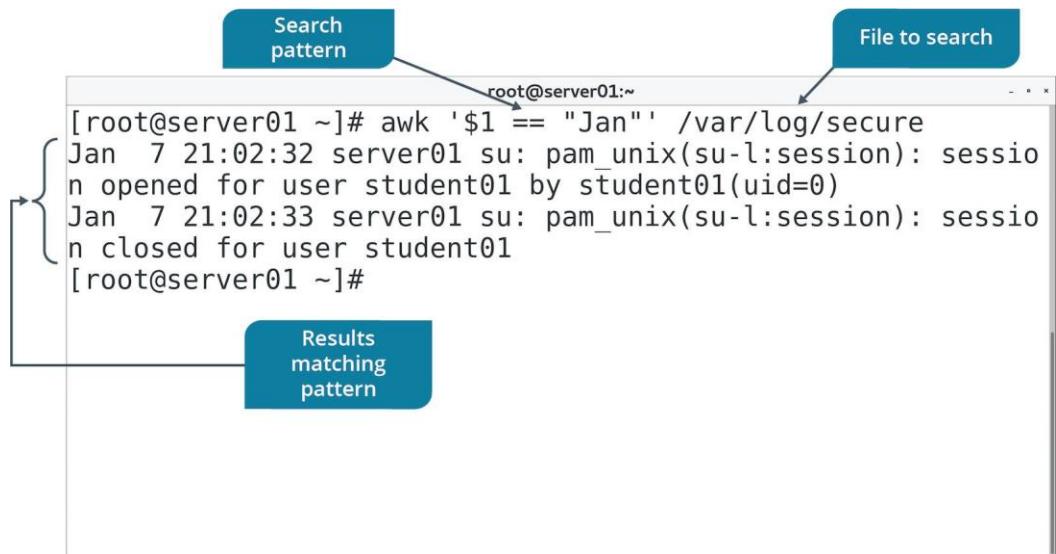
THE awk COMMAND

The **awk** command performs pattern matching on files. It is based on the AWK programming language. The **awk** keyword is followed by the pattern, the action to be performed, and the file name. The action to be performed is given within curly braces. The pattern and the action to be performed should be specified within single quotes. If the pattern is not specified, the action is performed on all input data; however, if the action is not specified, the entire line is printed. The **awk** command can be executed from the command-line or from within an **awk** script file.

The **awk** command can be used to process text files in a variety of ways, such as extracting text matching a certain pattern; deleting text matching a certain pattern; adding text matching a certain pattern; and much more.



Note: GNU's version of *awk* is called *gawk*



Searching a log for all entries recorded in January.

SYNTAX

The syntax of the `awk` command is `awk [options] ['patterns {actions}'] {file names}`

PATTERNS

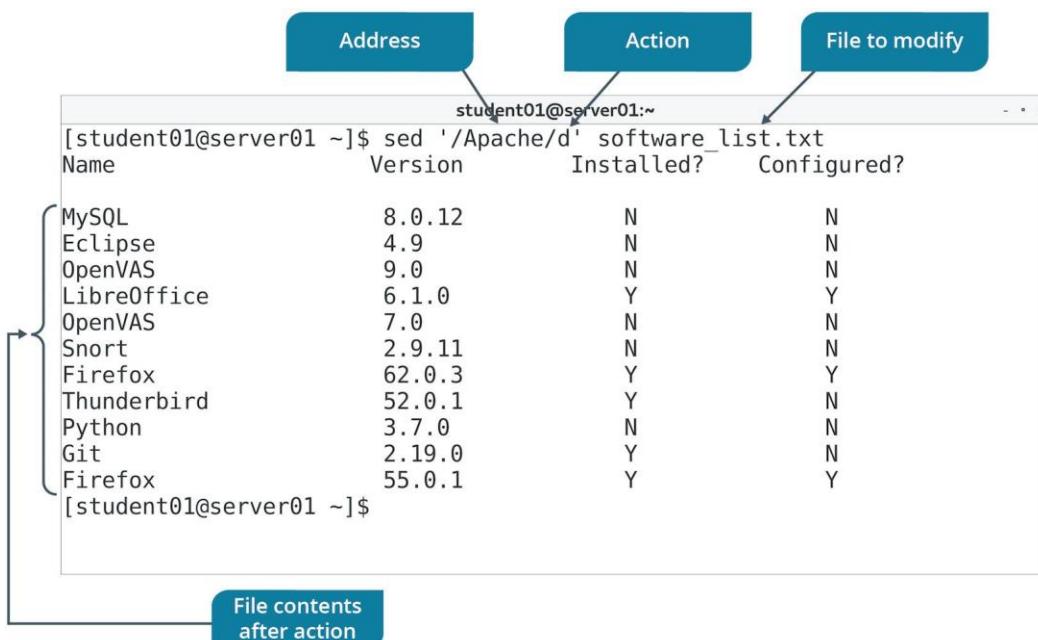
In `awk` scripts, you can provide patterns along with blocks of code. If a pattern matches any line in the input file, the code blocks in the script will be executed. The following table lists the types of patterns used.

Pattern	Description
<code>/regular_expression/</code>	Retrieves all the records beginning with "a", "b", or "c". Example: <code>/[abc]/</code>
<code>relational_expression</code>	Retrieves all the records whose first field contains the value "abc" Example: <code>\$1 == "abc"</code>
<code>pattern_1 && pattern_2</code>	Retrieves all the records whose first field contains the value "abc" and the second field contains the value "01" Example: <code>(\$1 == "abc") && (\$2 == "01")</code>
<code>pattern_1 pattern_2</code>	Retrieves records that satisfy the condition that the first field contains the value "abc" or the second field contains the value "01" or both. Example: <code>(\$1 == "abc") (\$2 == "01")</code>

Pattern	Description
pattern_1 ? pattern_2 : pattern_3	If the first field in a record contains the value "10", the fifth field is tested for its value. If the fifth record contains the value "20", then the record is printed. If the first field of a record does not contain the value "10", then the ninth field of the record is evaluated. If the ninth record contains the value "30", then the record is printed. Example: \$1 == "10" ? \$5 == "20" : \$9 == "30"
pattern_1, pattern_2	Prints a range of records, starting from the record whose first field contains the value "01". The records will be printed until the awk command finds a record whose first field contains the value "02". Example: \$1 == "01", \$1 == "02"

THE sed COMMAND

The **sed** or stream editor command is a program that you can use to modify text files according to various parameters. The **sed** command can also be used for global search and replace actions.



Deleting lines in a file that contain the term "Apache".

Some of the common command options and their uses are given in the following table.

Option	Used To
d	Delete the lines that match a specific pattern or line number.
-n,p	Print only the lines that contain the pattern.

Option	Used To
s	Substitute the first occurrence of the string in the file.
s,g	Globally substitute the original string with the replacement string for each occurrence in the file.

SYNTAX

The general syntax of the **sed** command is **sed {option/address/action}' {file names}**

Addresses tell **sed** to act only on certain lines or to act only on text that matches a given regular expression pattern. They are optional. Addresses are followed by the action to be performed when a match is found. The last argument is the name of the input file. The option, address, and action parameters are typically enclosed within single quotation marks.

THE **ln** COMMAND

The **ln** command is used to create a link to a file. Linking enables a file name in one directory (the link) to point to a file in another directory (the target). A link does not contain data of its own, only a reference to the target file. Any changes to the link will reflect in the target file. If you don't specify the link name, the **ln** command will create the link in your current working directory.

SYNTAX

The syntax of the **ln** command is **ln [options] {target name} [link name]**

ln COMMAND OPTIONS

The **ln** command has various options. Some of the frequently used options are given in the following table.

Option	Used To
--backup	Back up existing destination files.
-f	Remove existing destination files.
-s	Make symbolic links instead of hard links.
-i	Prompt to remove destination files.
-v	Print the name of a file before linking.

TYPES OF LINKS

Using the **ln** command, you can create two types of links: hard and symbolic (soft). Hard and symbolic links are a feature of the file system and are common in most file systems supported by Linux. The ext2, ext3, ext4, and XFS file systems all support hard and symbolic links.

A **hard link** is a reference to another file; it enables the file's data to have more than one name in different locations in the same file system. Applications treat a hard link as a real file. If the original file is deleted after a hard link is created, all its contents will still be available in the linked file. This is because the inode of a hard link is the same as its target; in other words, it points to the same object on the file system. Hard links cannot be created between two directories, nor can they be created between two files in different file systems.

A **symbolic link** is a reference to a file or directory that can span multiple file systems. If the original file or directory is deleted after a symbolic link is created, then the original content is lost. This is because the inode of a symbolic link is different than its target; in other words, it points to a different object on the file system. A symbolic link is also known as a soft link.



Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

EXAMPLES

The following is an example of creating a hard link using the `ln` command, where `/backup/backup-report` is the target of the link, and `~/backup-report` is the link itself:

```
ln /backup/backup-report ~/backup-report
```

The following is an example of the same, but creating a symbolic link instead of a hard link:

```
ln -s /backup/backup-report ~/backup-report
```

Activity 5-9

Discussing Processing Text Files

SCENARIO

Answer the following questions to check your understanding of the topic.

1. The echo command is a very powerful one indeed. It can be used in scripts or interactively at the command-line to do several different things. How can you use the echo command to display your name on the screen?
 2. How would you use the printf command to display your name and address on different lines using a single command?
 3. You have two files that contain more than 100 lines of text each. You need to compare them because, while they're supposed to be similar, checking each one against the other is time-consuming. Fortunately, Linux has a utility to compare the two files. How can you compare file1.txt and file2.txt using this utility?
 4. A user asks you to help him find a particular string ("audition") in a file, actors.txt. He can't seem to remember which actor he had auditioned for a commercial spot, and he needs to call her back. How can you find his actor using a simple command, rather than read the text file manually?
 5. You have a file that contains a very long list of activities (activities.txt) that were suggested by coworkers for things to do at the annual picnic. They are in no particular order, but you'd feel better if they were in alphabetical order, making them more readable. How can you rearrange the list into alphabetical order quickly?

Activity 5-10

Processing Text Files

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. You have a directory at `~/policies` that includes two text files, as well as the `software_list.txt` file in the home directory.

SCENARIO

Now that the software list and policy documents are all set and in the right locations, you can begin to analyze them more closely. In particular, you want to sort the software list so you can more quickly identify what software packages still need to be installed and/or configured. Likewise, you want to ensure that you know exactly what was changed from the first version of the AUP to the second version, so you don't have to read the entire thing from the beginning.

You also want to switch gears to your log analysis duties. You want to identify instances where users enter an incorrect password and fail to log in. This could point to users that are trying to access resources they are not authorized for. However, the authentication log can be very large, so you'll need to process it in order to extract only the relevant information.

1. Sort the software list file by name, then by which packages need to be installed and/or configured.
 - a) Enter `cd ~` to return to your home directory.
 - b) Enter `cat software_list.txt` to review the column structure of this file.
 - c) Enter `Sort -k1 software_list.txt`
 - d) Verify that the list was sorted by the first column, which is the name of each software package.
However, the sort operation was not perfect, as the column headers were included. There are several ways to stop this from happening, one of which you'll perform in a later topic.
 - e) Enter `sort -k3 software_list.txt` to sort by the "Installed?" column.
 - f) Sort by the "Configured?" column.
2. Retrieve the word count of the AUP files.
 - a) Enter `cd policies`
 - b) Enter `WC -w aup_v1.txt`
 - c) Verify that you can see the word count of version 1 of the AUP policy file.
 - d) Enter `WC -w aup_v1.txt aup_v2.txt`
 - e) Verify that you can see the word counts of both versions of the file, as well as a combined total.
3. Examine the differences between the AUP files.
 - a) Enter `diff aup_v1.txt aup_v2.txt`

- b) Verify that you are presented with the differences between each file, as well as suggested actions.

The differences are as follows:

- **33a34,41** means that after line 33 in the first file (version 1), lines 34–41 from the second file (version 2) need to be added in order for that chunk of text to be the same.
- The multiple **>** symbols indicate each line that must be added to the first file in order to be the same as the second file.
- In other words, the HR lead added this entire new section to version 2 of the policy.
- **35a44** means that after line 35 in the first file, line 44 from the second file needs to be added in order for the text to be the same.
- In other words, the HR lead added an entry to the revision history explaining her changes.

4. Search the authentication log for failed login attempts.

- a) Enter **sudo cat /var/log/secure**

- b) Verify that there are many entries in the authentication log.

Rather than read the entire log or search term-by-term for failure entries, you can use **grep** to bring all of the relevant information to the forefront in one command.

- c) Enter **SU - ariley** and provide an incorrect password to simulate an authentication failure.



Caution: Don't actually sign in.

- d) Enter **sudo grep failed /var/log/secure**

- e) Verify that you are presented with all lines in the log containing the text string "failed".

```
Dec 11 16:52:35 localhost unix_chkpwd[1254]: password check failed for user (root)
Dec 14 13:01:42 localhost unix_chkpwd[1739]: password check failed for user (root)
Dec 14 14:18:45 localhost unix_chkpwd[7351]: password check failed for user (ariley)
Dec 14 14:18:58 localhost sudo: student01 : TTY=pts/0 ; PWD=/home/student01 ; USER=root ; COMMAND=/bin/grep failed /var/log/secure
```

Activity 5-11

Linking Files

BEFORE YOU BEGIN

You are logged into the CLI as your student account. You have a `/backup/log` directory that you created earlier.

SCENARIO

You've decided to start organizing your backup directory, particularly with regard to log files. You want to create several subdirectories, each one a category that can pertain to the backed up logs. For example, you want to organize logs by type (e.g., authentication logs vs. app logs vs. kernel logs) and the year that they were generated. However, most logs can apply to multiple categories. Rather than have two or more distinct copies of each log, you decide to link these files together so that they're easier to manage.

You also want to be able to quickly access log backups from your home directory. So, you'll create a link in your home directory to a log in the backup directory.

1. Create new log backup directories and move the authentication log to one of them.
 - a) Enter `mkdir /backup/log/auth /backup/log/<year>` where `<year>` refers to the current year.
 - b) Enter `sudo cp /var/log/secure /backup/log/auth/secure`
 - c) Enter `cd /backup/log`
2. Create a hard link between the log files.
 - a) Enter `sudo ln auth/secure <year>/secure`
This creates a hard link to the file in the `auth` directory.
 - b) Enter `ls -l <year>` and verify that a file was created in the year directory.
 - c) Enter `sudo cat <year>/secure` and verify that its contents are the same as the authentication log.



Note: You can run `diff auth/secure <year>/secure` if you want to be sure.

3. Make a change in one file and see it reflected in the hard link file.
 - a) Enter `sudo nano auth/secure`
 - b) Press `Enter` to start a new line at the top.
 - c) Type `**BEGIN LOG ##-#####**` where `##-#####` is the current month and year.
For example: `**BEGIN LOG 01-2019**`

- d) Press **Ctrl+O** then **Enter** to save.
- e) Press **Ctrl+X** to quit.
- f) Enter **sudo head <year>/secure** and verify that the header you just added was also added to the hard link file.

```
[student01@localhost log]$ sudo head 2019/secure
**BEGIN LOG 01-2019**
Dec 11 13:31:02 localhost polkitd[843]: Loading rules from /etc/polkit-1/rules.d
Dec 11 13:31:02 localhost polkitd[843]: Loading rules from /usr/share/polkit-1/rules.d
Dec 11 13:31:03 localhost polkitd[843]: Finished loading, 0 rules
Dec 11 13:31:03 localhost polkitd[843]: Acquired the name org.freedesktop.PolicyKit1 on the system bus
Dec 11 13:31:12 localhost sshd[1346]: Server listening on 0.0.0.0 port 22
Dec 11 13:31:12 localhost sshd[1346]: Server listening on :: port 22
```

4. Remove one file and verify that the hard link file is still intact.
 - a) Enter **sudo rm auth/secure**
 - b) Enter **sudo cat <year>/secure** and verify that the hard link file's contents are still intact.
5. Attempt to create a link from your home directory to a log file in the backup directory.
 - a) Enter **Cd ~** to return to your home directory.
 - b) Enter **sudo ln /backup/log/<year>/secure auth-log**
 - c) Verify that the operation failed.
6. **Why did the system fail to create the link? What can you do to still create a link?**
7. Create a symbolic link to the log file.
 - a) Enter **ln -s /backup/log/<year>/secure auth-log**
 - b) Enter **sudo cat auth-log** and verify that your link has the expected log contents.
8. Delete the original log file and verify that the symbolic link was affected.
 - a) Enter **sudo rm /backup/log/<year>/secure**
 - b) Enter **sudo cat auth-log** and verify that no such file exists.
 - c) Enter **ls -l** and verify that the file is a broken link.

```
[student01@localhost ~]$ ls -l
total 4
1rwxrwxrwx. 1 student01 student01 23 Dec 14 14:22 auth-log -> /backup/log/2019/secure
drwxrwxr-x. 3 student01 student01 31 Dec 14 13:04 permissions-demo
drwxrwxr-x. 2 student01 student01 151 Dec 14 14:17 policies
-rw-r--r--. 1 student01 student01 816 Dec 14 14:12 software list.txt
```

The red text pointing to text with a black background indicates that the link is broken.

- d) Enter **rm auth-log** to delete the symbolic link.
-

Topic E

Manipulate File Output



EXAM OBJECTIVES COVERED

2.3 Given a scenario, create, modify, and redirect files.

All of this management and manipulation of files is useful for more than just looking at the results in a terminal. When you continue to use the terminal, or log out entirely, you'll want to ensure that some crucial information is stored in a file for later retrieval and analysis. You'll also benefit from using multiple commands in conjunction, making your administrative duties more efficient and powerful.

TEXT STREAMS

A **text stream** is a sequence of one or more lines of text that applications can leverage to read from or write to a particular device or system component. This enables the application to interface with components like the CLI, files, network sockets, and more, while hiding those components' details from the application.

In most Linux shells, there are three types of streams: standard input, standard output, and standard error.

STANDARD INPUT

Standard input, or **stdin**, is a text stream that acts as the source for command input. Standard input for the Linux command-line is usually generated from the keyboard. In the case of the GUI, the standard input can also come from the mouse.

STANDARD OUTPUT

Standard output, or **stdout**, is a text stream that acts as the destination for command output. By default, standard output from a Linux command is directed to the CLI.

STANDARD ERROR

Standard error, or **stderr**, is a text stream that is used as the destination for error messages. By default, the standard error stream prints error messages at the CLI.

INPUT/OUTPUT REDIRECTION

Redirection is the process of accepting input data from a source other than the keyboard and sending output data to a destination other than the display device. In other words, you can use redirection to bypass the default devices when working with input/output (I/O). Redirection is commonly used to accept input from files or send output to files using the stdin, stdout, and stderr streams.

REDIRECTION OPERATORS

There are several operators that are used to redirect input or output. These operators are described in the following table.

Operator	Used To	Example
>	Redirect the standard output to a file.	ls > file1.txt The output of the ls command will be redirected to a file named file1.txt .
>>	Append the standard output to the end of the destination file.	ls >> file1.txt The output of the ls command will be appended to a file named file1.txt .
2>	Redirect the standard error message to a file.	ls file3.txt 2> errorfile.txt Assuming that file3.txt does not exist, the resulting errors will not be displayed on the screen, but they will be redirected to a file named errorfile.txt .
2>>	Append the standard error message to the end of the destination file.	ls file3.txt 2>> errorfile.txt Assuming that file3.txt does not exist, the resulting errors will not be displayed on the screen, but they will be appended to a file named errorfile.txt .
&>	Redirect both the standard output and the standard error message to a file.	ls file1.txt file3.txt &> errorfile.txt Assuming that file1.txt exists and file3.txt does not, the resulting output and errors will not be displayed on the screen, but they will be redirected to a file named errorfile.txt .
<	Read the input from a file rather than from the keyboard or mouse.	mail user@address < myletter.txt The myletter.txt file will be taken as the input and attached to the email message.

Operator	Used To	Example
<code><<string</code>	Provide input data from the current source, stopping when a line containing the provided string occurs. When placed in a script, this is called a here document .	<pre>cat <<EOF This is a here document. EOF</pre> <p>The cat command will use the rest of the lines in this file as input. It will stop accepting that input when it reaches the string EOF. This string can be named anything you want. The output of the cat command would therefore be:</p> <pre>This is a here document.</pre>

PIPING

Piping is the process of combining the standard I/O streams of commands. It uses the standard output of one command as the standard input for another command. The output format of the first command should be compatible with the format that the second command works with. The pipe operator (`|`) can be used with most commands in Linux.



Commands being piped to other commands.

PIPING EXAMPLE

The `ls -l | grep audit` command mentioned earlier that searches for files named "audit" is an example of using a pipe. The standard output of the `ls -l` command is fed as standard input into the `grep audit` command, so that `grep` searches for the term within the directory listing.

THE xargs COMMAND

The `xargs` command reads from standard input and executes a command for each argument provided. Each argument must be separated by blanks. The pipe operator is used to make the output of the first command the input for the second command. The `xargs` command is commonly used with the `find` command to operate on each result that is found within the file or directory search.

The terminal window shows the following command sequence:

```
[root@server01 ~]# find ~ -type f -name "*.txt" | xargs chmod 775
[root@server01 ~]# ls -l | grep txt
-rwxrwxr-x. 1 root root 0 Jan 7 21:31 file1.txt
-rwxrwxr-x. 1 root root 0 Jan 7 21:31 file2.txt
-rwxrwxr-x. 1 root root 0 Jan 7 21:31 file3.txt
[root@server01 ~]#
```

Annotations explain the steps:

- Search for files matching pattern**: Points to the command `find ~ -type f -name "*.txt"`.
- New permissions**: Points to the command `chmod 775`.
- Change permissions for all files found**: Points to the pipe symbol `|` and the `xargs` command.

Changing the permissions for all files that match a given pattern.

SYNTA

X

The general syntax of the `xargs` command is `command [options] [arguments] | xargs [options] {command}`

EXAMPLE OF THE xargs COMMAND

Let's say you want to delete all of the files in the `/foo` directory that have a `.pdf` extension. You can use `xargs` to automate the process:

```
find /foo -type f -name "*.pdf" | xargs rm
```

The `find` command searches for all files in `/foo` that have a `.pdf` extension, then pipes the result to the `xargs` command. Because the results are delimited by a space, the `xargs` command will execute the `rm` command for each file in the results—removing all PDF files in the directory.

xargs COMMAND OPTIONS

The `xargs` command has various options.

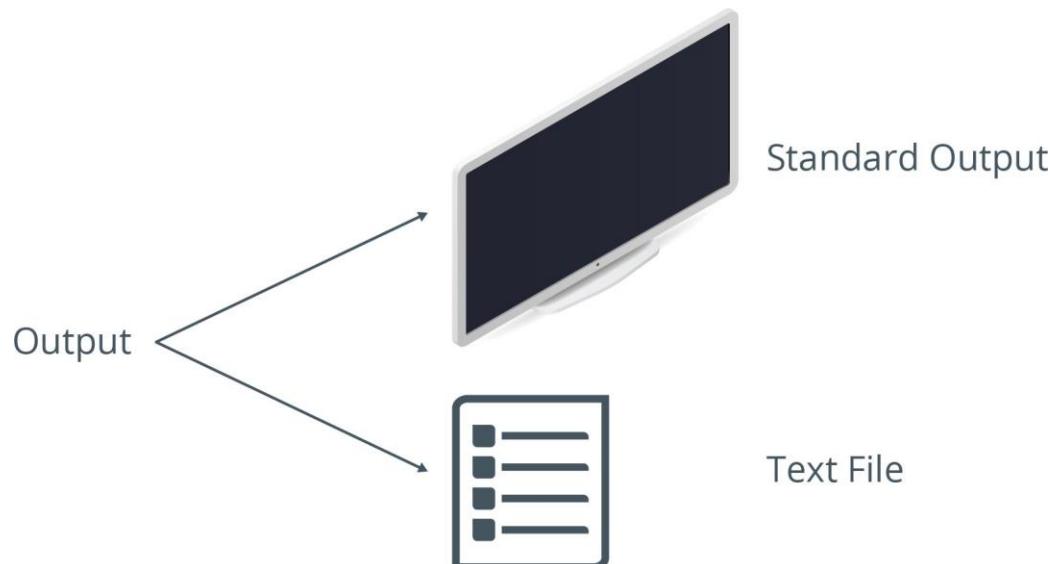
Option	Used To
<code>-I {replacement string}</code>	Consider each line in the standard input as a single argument.
<code>-L {number of lines}</code>	Read a specified number of lines from the standard input and concatenate them into one long string.
<code>-p</code>	Prompt the user before each command.
<code>-n {number of arguments}</code>	Read the maximum number of arguments from the standard input and insert them at the end of the command template.
<code>-E {end of string}</code>	Represent the end of the standard input.
<code>-t</code>	Write each command to the standard error output before executing the command.

Option	Used To
<code>-s {max size}</code>	Set the maximum allowable size of an argument list to a specified number of characters.

THE tee COMMAND

The `tee` command reads the standard input, sends the output to the default output device (the CLI), and also copies the output to each specified file. This command enables you to verify the output of a command immediately as well as store that output in a file for later reference. Like `xargs`, `tee` typically accepts input from another command using the pipe operator.

When used with the `-a` option, `tee` appends the output to each output file instead of overwriting it.



The `tee` command sends output to both the CLI and a text file.

SYNTAX

The general syntax of the `tee` command is `command [options] [arguments] | tee [options] {file names}`

EXAMPLE OF THE tee COMMAND

Let's say you want to check the contents of a directory and also output those contents to a file to process later. You could issue separate commands to do this, or you can use the `tee` command like so:

```
ls -l | tee listing.txt
```

THE /dev/null FILE

The `/dev/null` file, also known as the **null device**, is a file that discards all data written to it. Typically, you'd redirect an output stream to this file in order to confirm that the write operation was successful without actually writing to anything. This makes the file useful in testing commands, scripts, and other software. It is also useful

in suppressing error information in commands by redirecting error output (`2>`) to the `/dev/null` file.

TERMINAL REDIRECTION

A running process in Linux can be controlled by a terminal (CLI), and multiple terminals can run at once. Each controlling terminal is assigned an identifier. This identifier usually takes the format `/dev/tty#` where `#` is a number unique to that terminal.

You can redirect standard input and output to another controlling terminal by referencing its `/dev/tty` number. This can be useful when you need to redirect text streams between different running processes.

 **Note:** You can enter `tty` at the CLI to identify your controlling terminal.

 **Note:** To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 5-12

Discussing File Output

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **How would you create a file with the following content using a single command? Note that each component should be on its own line (a total of three lines), like addressing a letter in the mail:** Bob Jones
222
Main Street Dallas, TX

2. **Why does the following command send the name Sally Davis to the sally.txt file?** echo "Sally Davis" > sally.txt

3. **Your manager asks you to find the number of times that connections from IP address 10.0.10.5 have connected to server1. These events are typically logged in the /var/log/auth.log file. Using piping, how can you get this count for him?**

4. **What does the following command do?** cat file.txt > /dev/null

5. **What happens if you issue the following command?**
mail bbates@domain.com < addresses.txt

Activity 5-13

Manipulating File Output

DATA FILE

/opt/linuxplus/managing_files_and_directories/laptop_inv.txt

BEFORE YOU BEGIN

You are signed in to the CLI as your student account. You have a /backup directory.

SCENARIO

In the past, the IT team has kept an inventory of all laptops issued to employees. As part of the new roll-out, you'll need to copy this information to a document that will be stored on a Linux server. The source information isn't formatted very well, and isn't in any kind of useful order. So, you decide to create a new file from scratch. Afterward, you realize that the person who recorded the information made a mistake with the format of certain serial numbers. Instead of editing the file to replace every mistake individually, you'll leverage input and output redirection to fix the mistakes. Then, you'll output a sorted version that will be more useful for reference.

You also want to regularly check the contents of the backup directory and place the results in a continually updated file. You want to be able to see the results in real-time at the CLI as well, so you'll use the `tee` command to accomplish both.

Lastly, you'll use piping with `grep` to further hone your log analysis skills.

1. Use output redirection to start adding text to the laptop inventory file.
 - a) Enter `touch laptop_inv.txt` to create a blank file.
 - b) Enter `echo "User Make Serial No." > laptop_inv.txt`

Note: Separate each column by four spaces.



- c) Enter `cat laptop_inv.txt` and verify that the text output to the file.

2. Use output redirection to append text to the file.

- a) Enter `echo "jsmith Asus S489124" > laptop_inv.txt`
- b) Enter `cat laptop_inv.txt` and verify that the header was replaced by this new row.

This is because the `>` operator replaces any existing text with the provided string. You need to append that text.

- c) Reenter `echo "User Make Serial No." > laptop_inv.txt`

Note: Remember, you can press the **Up Arrow** to return to a command you previously entered.



- d) Enter `echo "jsmith Asus S489124" >> laptop_inv.txt`

Note: Again, separate each column by four spaces.



This time, you're using the append operator (`>>`).

- e) Verify that the file has both the header and the first row.

User Make Serial No.
jsmith Asus S489124

3. Use input redirection to replace all instances of a mistyped character in the file.

- a) Enter the following:

```
cp /opt/linuxplus/managing_files_and_directories/
laptop_inv.txt laptop_inv.txt
```

This will update your copy with a filled-in one.

- b) Examine the file and verify that the Asus serial numbers incorrectly start with the capital letter "S".

- c) Enter `tr S 5 < laptop_inv.txt`

- d) Verify that the instances of "S" were replaced with "5" and that the file was printed to the CLI.

4. Use both input and output redirection at the same time to create a new file with the corrections.

- a) Enter `tr S 5 < laptop_inv.txt > laptop_inv_fix.txt`

- b) Examine the corrected file and verify that the appropriate correction was made.

5. Use piping to sort the inventory list without the header.

- a) Enter `sort -k1 laptop_inv_fix.txt`

- b) Verify that, just like sorting the software list earlier, the header is included in the sort when it shouldn't be.

- c) Enter `tail -n +3 laptop_inv_fix.txt | sort -k1`

The `tail -n +3` command outputs everything after and including the third line, which is when the header ends. You are piping the output of this command to the `sort` command, which takes it as input.

- d) Verify that the inventory is now sorted by user name, but does not include the header.

hroberts	Lenovo	8989090
jcook	Dell	1489284
jsmith	Asus	5489124
kriley	Dell	1390390
lbarnes	Asus	5393892
manderson	Acer	2988481
mstephens	Asus	5737481
nporter	Asus	5892849
rburton	Lenovo	8139003
sarmstrong	Dell	1923843
tlee	Lenovo	8112091
tramirez	Acer	2942349

6. Use the **tee** command to redirect output to both the CLI and a file at the same time.

- a) Enter **sudo ls -IR /backup > backup_report**
- b) Verify that **ls** didn't print its results to the CLI.
- c) Enter **sudo ls -IR /backup | tee backup_report**
- d) Verify that **ls** did print its results to the CLI.

This is because piping the **ls** command to **tee** instead of doing a **stdout** redirect ensures that the results will appear at both the CLI and the specified file.

- e) Examine the **backup_report** file and verify that it also lists directory information.

7. Use **grep** and **cut** together to make log analysis easier.

- a) Enter **sudo grep 'password check failed' /var/log/secure**

This prints all instances of the text "password check failed" from the authentication log. However, it also prints every single part of the line, much of which isn't relevant and just adds to the noise.

- b) Enter **sudo cut /var/log/secure -d " " -f5-12**

The **cut** command, using the **-d** option, trims each line using a space as a delimiter.

The **-f5-12** option specifies the range of the delimiter to extract. So, you're only extracting approximately the middle chunk of each line. However, you're still seeing every line of the log.

- c) Enter **sudo grep 'password check failed' /var/log/secure | cut -d " " -f5-12**

- d) Verify that you extracted all lines matching the provided string, as well as only the portion of the line that is relevant to your needs.

The results show the system function that was called, an explanation of the event, as well as the user the event applies to.

Summary

In this lesson, you managed files and directories by creating, editing, searching, and manipulating them in various ways. These skills will enable you to work with your files more efficiently and productively.

What are some ways that leveraging the power of text streams and standard I/O could enhance your administrative tasks on Linux systems?

Which Linux text editor do you prefer to use? Why?



Practice Question: Additional practice questions are available on the course website.

Lesson 6

Managing Kernel Modules

LESSON TIME: 1 HOUR

LESSON INTRODUCTION

One of the defining features of Linux® is that it is modular, enabling you to adjust low-level system configurations at boot and during operation. This provides you with a great deal of flexibility as far as adjusting how your system runs and what types of devices it can leverage. In this lesson, you'll explore more about the Linux kernel and some of its features, and then you'll customize the kernel to meet your unique business needs.

LESSON OBJECTIVES

In this lesson, you will:

- Identify the role and functions of the Linux kernel.
- Install and configure kernel modules.
- Monitor kernel modules.

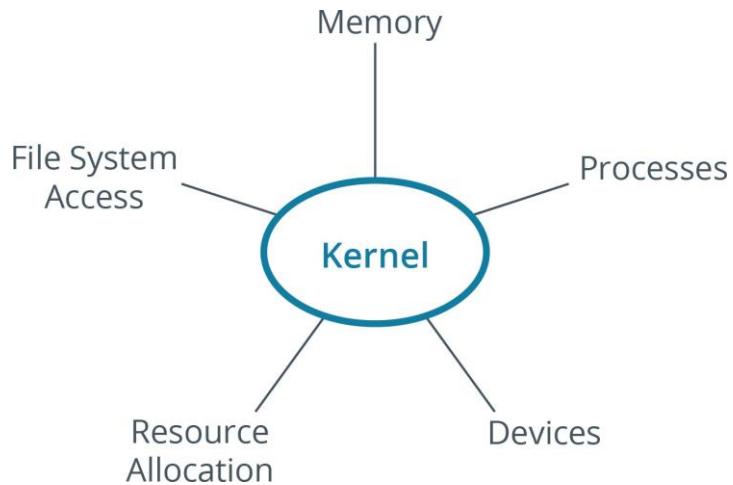
Topic A

Explore the Linux Kernel

You'll begin by identifying some of the key concepts and components that make up the Linux kernel. This will give you a better sense of what services the kernel provides and how you might go about customizing those services.

KERNEL

The **kernel** is the core of an operating system. All other components rely on it. The kernel manages file system access, memory, processes, devices, and resource allocation on a system. The kernel also controls all the hardware devices plugged into the system. It is one of the first elements to be loaded on startup and remains in the main memory during the computer's operation. The kernel also contains system-level commands and other functions that are normally hidden from users.



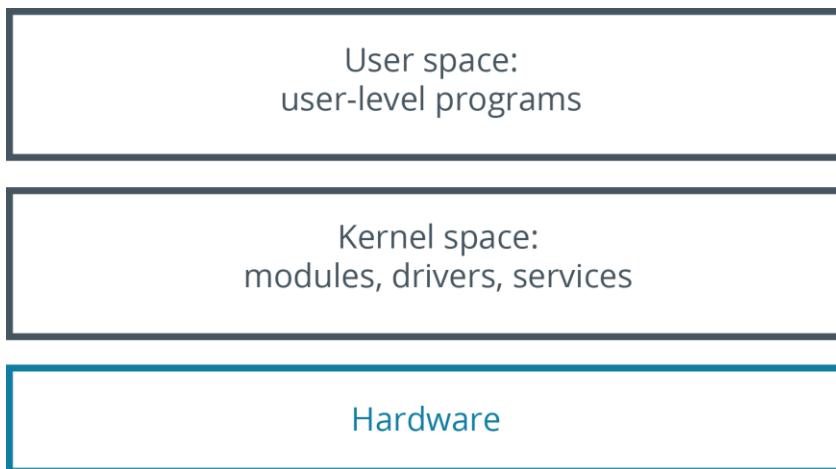
The elements managed by a kernel.

KERNEL SPACE AND USER SPACE

Kernels tend to divide software running in memory into two spaces: kernel space and user space. The **kernel space** is simply where the kernel executes the services that it provides. The **user space** is the area of memory that includes everything outside of kernel space. This can include everything from high-level applications that the user interacts with directly, to processes that run in the background, to various low-level system libraries.

Software running in user space is able to access resources provided by kernel space through the use of system calls. These calls provide a user space application with the resources it needs to perform a task. For example, an application might issue a system call to the kernel so that it can leverage input/output (I/O) services that write data to a storage device.

The split between these two memory regions is useful because it promotes greater stability and security. Software in one space cannot necessarily interfere with software in the other.



The different kernel spaces.

TYPES OF KERNELS

Kernels can be classified as monolithic or microkernel. In a **monolithic kernel**, all system modules, such as device drivers or file systems, run in kernel space. As a result, a monolithic kernel can interact quickly with devices. However, its main disadvantage is its size, which leads to higher consumption of RAM. In addition, a failure in a device driver can lead to system instability in a monolithic kernel.

In a **microkernel** architecture, the kernel itself runs the minimum amount of resources necessary to actually implement a fully functional operating system. Compared to monolithic kernels, microkernels have smaller kernel spaces and instead have larger user spaces. This means microkernels are smaller in overall size and consume less memory. In addition, they are typically more stable. However, microkernels tend to offer worse performance than monolithic kernels.

DEVICE DRIVERS

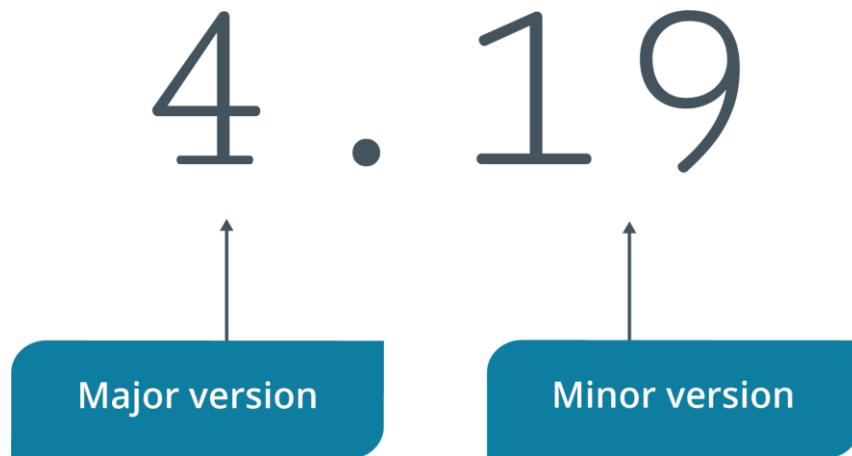
A **device driver** is a software program that enables a computer's operating system to identify the characteristics and functions of a hardware device, communicate with it, and control its operations. It acts as an interface between the operating system and hardware devices such as storage drives, printers, scanners, monitors, and keyboards. Device drivers can be included in the operating system or installed on demand.

THE LINUX KERNEL

The **Linux kernel** is a free and open source monolithic kernel that manages all other resources on the operating system. As a monolithic kernel, device drivers run within kernel space and have full access to hardware. The architecture of the Linux kernel provides many useful features, including virtual memory management, support for TCP/IP networking, shared libraries, and many more.

One important quality of the Linux kernel is its modularity. This enables users to configure and extend kernel functionality to meet their needs.

The Linux kernel is continually updated by creator Linus Torvalds and many other volunteers. Each new version of the kernel is given a kernel version number to distinguish it from past and future versions. The current naming convention for kernel versions is **major.minor** where **major** is the major version number and **minor** is the minor version number. For example, version 4.19 was released in October 2018.



The modern versioning format for the Linux kernel.

KERNEL VERSION HISTORY

For versions 2.6.39 and prior, the kernel number format was **W.X.Y.Z** where **W** is the major version number, **X** is the major revision number, **Y** is the minor revision number, and **Z** is the patch number.

After 2.6.39, Torvalds decided to shorten the version number format, and the next version number was 3.0. After 3.19, rather than proceed to 3.20, Torvalds decided to jump to 4.0. This was for readability purposes, not due to any technical advances. Newer versions of the kernel will continue this trend of avoiding large minor numbers.

THE uname COMMAND

By default, **uname** prints the name of the kernel—Linux. You can view the kernel version number of your current system by using the **uname -r** command. You can also enter **uname -i** to view the hardware platform. To print all information, enter the **uname -a** command.

The screenshot shows a terminal window with the command [root@server01 ~]# uname -a. The output is: Linux server01 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux. Three teal-colored rounded rectangular boxes with arrows point to specific parts of the output: one points to "Kernel name" (root@server01), another to "Kernel version" (3.10.0-862.el7.x86_64), and a third to "Hardware platform" (x86_64 GNU/Linux).

Viewing kernel information.

KERNEL LAYERS

The kernel performs various functions to control and manage the operations of a system. It is composed of several layers that operate in kernel space.

Kernel Layer	Function
System Call Interface (SCI)	<p>Handles system calls sent from user applications to the kernel. This enables user space applications to request services from the kernel space, like processing time and memory allocation.</p> <p>This layer also enables the kernel to schedule and process system calls and manage multiple system calls simultaneously.</p>
Process management	<p>Handles different processes by allocating separate execution space on the processor and ensuring that the running of one process does not interfere with other processes.</p> <p>Through scheduling, the kernel implements sharing of processor time for executing multiple processes.</p>
Memory management	<p>Manages the computer's memory, which is one of the complex tasks performed by the kernel. Like processor sharing, the system's memory also needs to be shared among different user space resources.</p> <p>The kernel maps or allocates the available memory to applications or programs on request and frees the memory automatically when the execution of the programs is complete, so that it can be allocated to other programs.</p>
File system management	<p>Manages the filesystem, which involves storing, organizing, and tracking files and data on a computer.</p> <p>The kernel also supports a virtual file system (VFS) that provides an abstract view of the underlying data that is organized under complex structures, so that it appears to be a single structure.</p>
Device management	<p>Manages devices by controlling device access and interfacing between user applications and hardware devices of the computer.</p> <p>When a user space application sends a system call, the kernel reads the request and passes it on to the drivers that manage the activities of that particular device.</p>

Activity 6-1

Discussing the Linux Kernel

SCENARIO

Answer the following questions to check your understanding of the topic.

1. In the monolithic kernel vs. microkernel debate, Linus Torvalds chose a monolithic kernel for Linux. Why?
 2. As an administrator, it is one of your jobs to keep track of the kernel versions on your Linux systems and attempt to be consistent where possible. Which command can you run to see the kernel version?
 3. One of the most important qualities of the Linux kernel is that it is modular. What does this feature do for Linux as a whole?
 4. How does the Linux kernel manage devices?
 5. How does the Linux kernel handle memory management for applications?

Activity 6-2

Exploring the Linux Kernel

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

As a system administrator, you may need to troubleshoot issues related to the kernel. So, you want to explore kernel concepts to refresh your knowledge.

1. View version information about the currently running kernel.
 - a) Enter `uname -a` to view information related to the currently running Linux kernel.

2. **What is the base version of your currently running kernel according to the `uname` command?**
 - 2.4
 - 2.6
 - 3.4
 - 3.10
 - 4.18

3. **True or false? According to the `uname` command, you are running a 32-bit hardware platform.**
 - True
 - False

4. **Which function is associated with the SCI layer of the kernel?**
 - Passing requests to device drivers.
 - Sending service requests to the kernel.
 - Allocating processor time for functions.
 - Processing scheduling functions.
 - Organizing files on the file system.

5. What are the major functions performed by the kernel? (Choose two.)

- Kernel initialization
- Process management
- Memory management
- Module installation
- Dependency management

6. Which of the following accurately describe the user space? (Choose two.)

- It is the area of the memory where the kernel executes its services.
- It is the area of memory in which most high-level software runs.
- It is the part of the system that only logged in users can access.
- It is the area of memory in which background processes and low-level system libraries run.

7. What is one disadvantage of a monolithic kernel compared to a microkernel?

- Monolithic kernels are slower to access devices.
- Monolithic kernels are larger and consume more RAM.
- Monolithic kernels have a smaller kernel space and are less extensible.
- Monolithic kernels can only run the bare minimum software to qualify as a fully functional OS.

8. True or false? The Linux kernel is modular, enabling users to extend its functionality.

- True
 - False
-

Topic B

Install and Configure Kernel Modules



EXAM OBJECTIVES COVERED

- 1.2 Given a scenario, install, configure, and monitor kernel modules.
- 1.3 Given a scenario, configure and verify network connection parameters.
- 4.1 Given a scenario, analyze system properties and remediate accordingly.

Now that you've explored the Linux kernel, you're ready to install and configure some kernel modules. This will enable you to extend the kernel's functionality for specific purposes.

KERNEL MODULES

A **kernel module** is a system-level object that extends the functionality of the kernel. It can be dynamically loaded into the kernel or unloaded from the kernel when required. It enables the kernel to update or recompile itself without requiring the system to reboot.

The advantages of kernel modules are:

- They reduce the burden on the kernel because otherwise all of the modules' functionality would have to be added directly to the kernel.
- Dynamic loading of kernel modules facilitates lower memory consumption.
- They avoid having to rebuild and reboot the system when new functionality is required.

Kernel module file consists of a .ko extension. Modules built for a specific kernel version may not be compatible with another version of the kernel.

THE /usr/lib/ DIRECTORY

The **/usr/lib/** directory contains shared libraries and binaries for general programs and software packages. The files in this directory are not meant to be executed by the user or custom shell scripts. More specifically, the **/usr/lib/ modules/** directory contains the modules of different kernel versions that are installed. It holds a directory named after the kernel's version number. Inside this directory, modules are stored across various subdirectories based on the categories they belong to. For example, a Bluetooth® driver may be stored in:

`/usr/lib/modules/<kernel version>/kernel/drivers/bluetooth/`



Note: These modules may also be available from the `/lib/modules/` directory.

KERNEL MODULE SUBDIRECTORIES

Inside `/usr/lib/modules/<kernel version>/kernel/` are several subdirectories, some of which are described in the following table.

Director	Contains Modules For
arch	Architecture-specific support.

Director	Contains Modules For
crypto	Encryption and other cryptographic functions.
drivers	Various types of hardware.
fs	Various types of file systems.
net	Networking components such as firewalls and protocols.

KERNEL MODULE MANAGEMENT COMMANDS

Kernel module management commands enable you to view, load, unload, or modify kernel modules.

Command	Used To
lsmod	Display the currently loaded kernel modules, their sizes, usage details, and their dependent modules.
modinfo	Display information about a particular kernel module, such as the file name of the module, license, description, author's name, module version number, dependent modules, and other parameters or attributes. The syntax of this command is modinfo [options] {module name}
insmod	Install a module into the currently running kernel. This command inserts only the specified module and does not insert any dependent modules. The syntax of this command is insmod {module name}
rmmod	Remove a module from the currently running kernel. The syntax of this command is rmmod {module name}

THE modprobe COMMAND

The **modprobe** command is used to add or remove modules from a kernel. This command is capable of loading all the dependent modules before inserting the specified module. It is therefore preferred over using the **insmod** and **rmmod** commands.

To add modules using **modprobe**, use the **-a** option and specify the modules you want to add. To unload a module, use the **-r** option and specify the modules you want to remove.

SYNTAX

The syntax of the **modprobe** command is **modprobe [options] [module names]**

modprobe COMMAND OPTIONS

In addition to options for adding and removing modules, the **modprobe** command has more options.

Option	Used To
-f	Force the module to be inserted or removed.
-n	Conduct a dry run, i.e., output results without actually executing operations.
-s	Print errors to the system log (syslog) rather than stderr.

Option	Used To
-v	Enable verbose mode.

THE depmod COMMAND

In order for **modprobe** to accurately install dependent modules, it reads the **modules.dep** file to identify how modules are linked to one another. The **depmod** command is used to update this database of dependencies so that **modprobe** can function properly.

The **depmod** command searches the contents of **/lib/modules/<kernel version>/** for each module. A module may export a "symbol", indicating that it can provide a service to other modules. Other modules may call these exported symbols in their own code to leverage their capabilities. So, **depmod** builds the **modules.dep** file by aggregating all instances of symbols being exported and used.



*Note: The **modules.dep** file is located in the **/usr/lib/modules/<kernel version>/** directory.*

SYNTAX

The syntax of the **depmod** command is **depmod [options]**

MORE ON SYMBOLS

Symbols provide a way for modules to call upon the functions or other programming objects of other modules. For example, **module1** has a C function named **foo()** that performs some useful task. Another module, **module2**, wants to use **foo()** when it is linked to the kernel, rather than incorporate that routine in its own code. This is only possible if **module1** explicitly exports **foo()** for external use. It does this by using **EXPORT_SYMBOL()** or one of its variants on the function. The **foo()** function then becomes available as a symbol for any other module in the kernel to leverage.

KERNEL MODULE CONFIGURATION

The **/etc/modprobe.conf** file is a configuration file that contains settings that apply persistently to all the modules loaded on the system. It is used to configure modules and their dependencies and also specify module aliases. An alias is just an alternative name to use for a module.

In newer Linux distros, this file is deprecated. The **/etc/modprobe.d/** directory is used instead, and contains various **.conf** files. Other than creating aliases, these files can tell **modprobe** to run additional modules with specific options when your chosen module is loaded into the kernel. This enables the chosen module to leverage another module's functionality without actually loading it into the kernel. You might do this when your module doesn't directly depend on a second module, but does run better if that second module is installed.

```
root@server01:~#
[root@server01 ~]# cat /etc/modprobe.d/btusb.conf
alias blue btusb
```

Module configuration

Configuration file

An alias configured for a specific kernel module.

CONFIGURATION FILE COMMANDS

Files ending in .conf in the /etc/modprobe.d/ directory can use one of several commands.

Command	Used To
alias {alternative name} {module name}	Specify an alternative name for a module with a long name.
blacklist {module name}	Ignore internal aliases, which occur when modules define their own aliases.
install {module name} {command}	Run the specified command without inserting the module into the kernel.

KERNEL PARAMETERS

In addition to loading modules into the kernel at runtime, you can also change some of the kernel's parameters while it is running. You can use these parameters to improve system performance, harden security, configure networking limitations, change virtual memory settings, and more.

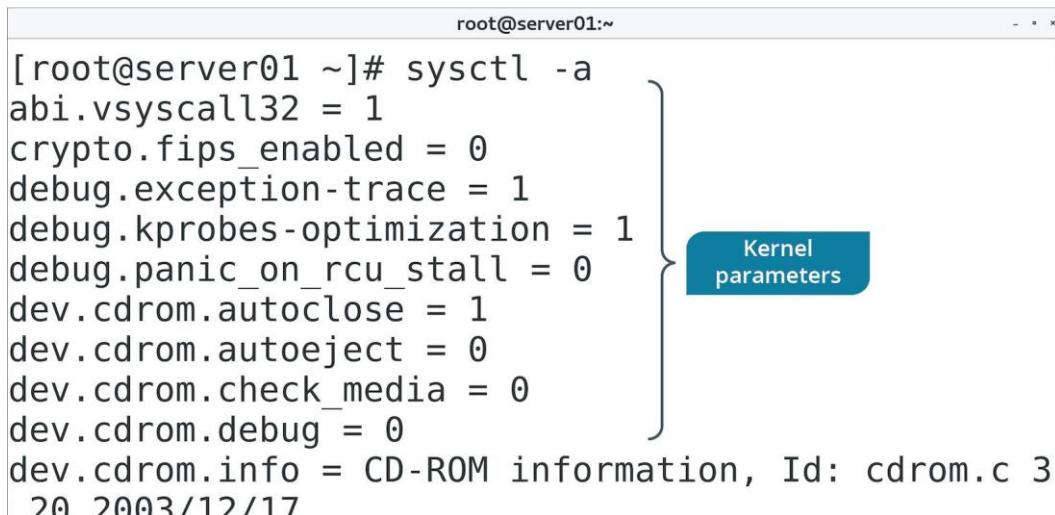
The **/proc/sys/** directory lists the parameters that you can configure on your system. Like the directories containing kernel modules, this **/proc/sys/** directory is divided into several categories, including the following.

Director	Includes Parameters Related To
crypto	Encryption and other cryptographic services.
debug	Debugging the kernel.
dev	Specific hardware devices.
fs	File system data.
kernel	Miscellaneous kernel functionality.
net	Networking functionality.
user	User space limitations.
vm	Virtual memory management.

THE **sysctl** COMMAND

The **sysctl** command is used to view or set kernel parameters at runtime. It has various options, as defined in the following table.

Option	Used To
-a	Display all parameters and their current values.
-w {parameter}={value}	Set a parameter value.
-p[file name]	Load sysctl settings from the specified file, or /etc/sysctl.conf if no file name is provided.
-e	Ignore errors about unknown keys.
-r {pattern}	Apply a command to parameters matching a given pattern, using extended regular expressions.



```
root@server01 ~]# sysctl -a
abi.vsyscall32 = 1
crypto.fips_enabled = 0
debug.exception-trace = 1
debug.kprobes-optimization = 1
debug.panic_on_rcu_stall = 0
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3
20 2003/12/17
```

Displaying kernel parameters.

SYNTA

X

The syntax of the **sysctl** command is **sysctl [options]**

THE **/etc/sysctl.conf** FILE

The **/etc/sysctl.conf** file enables configuration changes to a running Linux kernel. These changes might include improvements to networking, security configurations, or logging of information.

Note: To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.



Activity 6-3

Discussing Installing and Configuring Kernel Modules

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You're sure that you compiled and loaded the kernel module for your new Ethernet card, but the card doesn't work. How can you check to be sure the kernel module is loaded?

 2. You have downloaded a new kernel module that you want to load into your running system's kernel. You do so using the `insmod` command. You checked, and the kernel module is in the list of loaded modules, but it isn't working. What might have gone wrong?

 3. You loaded a kernel module that has caused your system to become unstable. How do you remove the errant module?

 4. The developers at your company have created custom kernel modules to optimize in-house application performance. They supply you with all the necessary files to load the new modules, including dependent modules. When you load the first primary module using `modprobe` you receive multiple errors and the modules don't load because of broken dependencies. Which file or files can you examine to find the issues?

 5. Your manager comes to you to discuss performing some no-cost security hardening on your Linux systems. Specifically, she asks you to protect the kernel from attacks. Which command and associated configuration file can you investigate to assist you in this task?
-

Activity 6-4

Installing and Configuring Kernel Modules

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

You want to be able to wirelessly transfer files from the Linux server to your mobile device. So, you purchase a USB Bluetooth adapter and plug it into an available port on the server. However, you can't get the adapter to work properly. After examining the system, you discover that the driver for USB Bluetooth is not available. So, you'll inspect the kernel and see if you can identify and load the module that enables this functionality.

1. Examine what modules are currently running.

- a) Enter **lsmod | less**
- b) Briefly scan through the list of installed kernel modules.
- c) Press **q** to quit.
- d) Enter **lsmod | grep bluetooth**
- e) Verify that there are no results.

You don't yet know the name of the relevant module, so this isn't necessarily definitive proof that it isn't loaded.

2. Search for the appropriate module.

- a) Enter **uname -r** to retrieve the kernel version of the system.
- b) Enter **cd /lib/modules/<kernel version>/kernel/drivers**



Note: Remember to use tab completion to fill the kernel version automatically.

- c) Enter **ls | grep bluetooth** and verify that there is a **bluetooth** directory.
- d) Enter **cd bluetooth**

- e) Enter **ls** to see the available Bluetooth driver modules.

```
[student01@localhost bluetooth]$ ls  
ath3k.ko.xz      btintel.ko.xz      btusb.ko.xz  
bcm203x.ko.xz    btmrvl.ko.xz     hci_uart.ko.xz  
bfusb.ko.xz      btmrvl_sdio.ko.xz  hci_vhci.ko.xz  
bpa10x.ko.xz     btrtl.ko.xz  
btbcm.ko.xz      btsdio.ko.xz
```

3. Do any of these look like they could be a driver for a USB device that can send and receive Bluetooth signals?

4. Learn more about this module.

- Enter **modinfo btusb.ko.xz | less**
- Read the information about this module, noting the following:
 - The description indicates that this is a generic Bluetooth USB driver.
 - It has many different aliases that aren't very user friendly.
 - It depends on several other modules.
- Press **q** to quit.

5. Configure an alias for the Bluetooth USB module.

- Enter **cd /etc/modprobe.d**
- Enter **sudo vim btusb.conf** to create a configuration file for the module.
- Using Vim, type **alias blue btusb** as the first line.
- Save and close the file.

6. Insert the Bluetooth USB module into the running kernel.

- Enter **sudo depmod** to update the dependencies database.
- Enter **sudo modprobe -a blue**
- Enter **lsmod | grep btusb**
- Verify that the **btusb** module is listed, indicating that it is inserted into the kernel.

7. Notice that there are other modules that begin with bt, as well as a module called bluetooth. Why were these added to the kernel as well?

Topic C

Monitor Kernel Modules



EXAM OBJECTIVES COVERED

- 1.2 Given a scenario, install, configure, and monitor kernel modules.
- 2.7 Explain the use and operation of Linux devices.

After you install and configure kernel modules, it's a good idea to monitor those modules. In this topic, you'll verify that the modules you installed were actually loaded into the kernel, and that any configurations you made were implemented properly.

THE /proc/ DIRECTORY

The `/proc/` directory is a virtual file system (VFS) that provides significant information about the kernel's running process. Some of the files in the `/proc/` directory are listed in the following table.

File	Contains
<code>/proc/cmdline</code>	Options passed to the kernel by the boot loader at boot time, such as mounting the kernel as read-only.
<code>/proc/cpuinfo</code>	CPU information, such as its architecture, name, clock speed, cache size, and more.
<code>/proc/devices</code>	A list of character and block device drivers loaded into the currently running kernel.
<code>/proc/filesystems</code>	A list of file systems types that are supported by the kernel, as well as if any are currently mounted.
<code>/proc/meminfo</code>	Information about RAM usage, including total memory, free memory, and much more.
<code>/proc/modules</code>	Information about modules currently installed on the system. An alternative to the <code>lsmod</code> command.
<code>/proc/stat</code>	Various statistics about the system since it was last rebooted.



Note: Recall that `/proc/mounts` and `/proc/partitions` are also available here.

THE /proc/version FILE

The `/proc/version` file specifies several points of information about the Linux kernel:

- The version of the Linux kernel currently running.
- The version of the **GNU Compiler Collection (GCC)** used to compile the kernel.
- The user name of the kernel compiler.
- The time the kernel was compiled.

The version of the kernel may impact system functionality, so you can use this file to validate that version.

```
root@server01 ~]# cat /proc/version
Linux version 3.10.0-862.el7.x86_64 (builder@kbuilder.dev.centos.org) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-28) (GCC) ) #1 SMP Fri Apr 20 16:44:24 UTC 2018
```

Kernel version GCC version Time of compile Compiler user name

The `/proc/version` file.



Note: Remember, you can also use the `uname -r` command to retrieve the kernel version.



Note: GCC originally stood for GNU C Compiler because it only supported the C programming language. It now works with several programming languages.

THE dmesg COMMAND

The `dmesg` ("display message" or "driver message") command is used to print any messages that have been sent to the kernel's message buffer during and after system boot. Device drivers send messages to the kernel indicating the status of modules and parameters that the drivers interface with. These drivers can also send diagnostic messages to the kernel in case they encounter errors. Other kernel components can also send messages to the buffer.

Driver messages

```
[root@server01 ~]# dmesg -H
[Jan 2 18:40] microcode: microcode updated early to revision
[ +0.000000] Initializing cgroup subsys cpuset
[ +0.000000] Initializing cgroup subsys cpu
[ +0.000000] Initializing cgroup subsys cpuart
[ +0.000000] Linux version 3.10.0-862.el7.x86_64 (builder@kbuilder.dev.centos.org)
[ +0.000000] Command line: BOOT_IMAGE=/vmlinuz-3.10.0-862.
[ +0.000000] e820: BIOS-provided physical RAM map:
[ +0.000000] BIOS-e820: [mem 0x0000000000000000-0x0000000000000000]
[ +0.000000] BIOS-e820: [mem 0x00000000000058000-0x00000000000059000]
[ +0.000000] BIOS-e820: [mem 0x00000000000059000-0x0000000000008c000]
[ +0.000000] BIOS-e820: [mem 0x000000000000f0000-0x000000000000]
[ +0.000000] BIOS-e820: [mem 0x000000000000100000-0x000000000000]
```

Displaying driver messages.

In addition to using the `dmesg` command, you can also access the message buffer from the `/var/log/dmesg` file. In either case, you can leverage `dmesg` to look for potential issues with kernel components or to validate that certain modules are being loaded.

SYNTAX

The syntax of the `dmesg` command is `dmesg [options]`

dmesg COMMAND OPTIONS

You can use various options with the `dmesg` command.

Option	Used To
<code>-c</code>	Clear the kernel buffer after printing its contents.
<code>-f {facility list}</code>	Restrict output to the specified comma-separated list of facilities. A facility is a component category that is producing messages, such as <code>user</code> for user-level messages.
<code>-l {level list}</code>	Restrict output to the specified comma-separated list of levels. A level defines a message's nature and priority, such as <code>notice</code> for messages that aren't considered critical.
<code>-e</code>	Display a human-readable version of the time of each message as well as its delta, or the difference in time between subsequent messages.
<code>-L</code>	Color-code messages for easier readability.
<code>-H</code>	Output in a human-friendly format, combining both <code>-e</code> and <code>-L</code> options and using a text pager.
<code>-h</code>	List the available options, as well as the available facilities and levels.

 **Note:** To learn more, check the **Video** tab on the course website for any videos that supplement the content for this lesson.

Activity 6-5

Discussing Monitoring Kernel Modules

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You're a new system administrator and you might not know all the commands to find out every detail about a Linux system. Although you don't know the commands, where can you go on the file system to find out the same information in text file format?
 2. A junior system administrator asks for your help in determining which components make up the Linux server he's working on. He needs to know brands, models, and other details of certain components to buy replacement parts. To which command or file(s) can you direct him?
 3. You believe that the network interface card is failing on one of your servers and you need to get the specifics of it so that you can replace it with the same product. Without going to the data center and opening the server's case, how can you find the network card's details?
 4. Your Linux web server seems to have a memory problem that you can't pinpoint through checking web logs or system logs. Where can you look to find details about system memory usage?
 5. You need to know which file systems are supported by one of your Linux systems. How can you find out?

Activity 6-6

Monitoring Kernel Modules

BEFORE YOU BEGIN

You are logged in to the CLI as your student account. Previously, you installed a USB Bluetooth driver module in the kernel.

SCENARIO

Now that you installed the USB Bluetooth module, you want to make sure it was successfully loaded by the kernel and that there are no errors. You also want to identify your kernel version details in case you need to reference it during troubleshooting.

1. Enter `cat /proc/version` and use the result to answer the following questions.

When was the kernel last compiled?

What version of the GCC is your kernel running?

Why might this information be useful?

2. Examine the kernel message buffer.
 - a) Enter `dmesg -h`
 - b) Note the different facilities and log levels available. Examples include `warn`, `err`, `notice`, etc.
 - c) Enter `dmesg -H`
 - d) Verify that you can navigate through many pages of kernel messages.
Not all of the information here will be useful to you, so you'll need to filter what you're looking for.
 - e) Press `q`.
3. Filter the kernel message buffer for more useful messages.
 - a) Enter `dmesg -H -I warn`
 - b) Verify that the results have been filtered.
All of these messages are marked as warning conditions. These don't necessarily indicate errors, but call attention to behavior that might be worth checking.

- c) If necessary, press **q**.
 - d) Enter **dmesg -H -l err**
- These messages *do* indicate errors. You might not have any results, which means the kernel hasn't recorded any errors thus far.
- e) If necessary, press **q**.

4. Search the kernel message buffer for evidence of USB drivers being loaded.

- a) Enter **dmesg -H | grep usb**
- b) Examine the results.

The kernel records when USB storage devices are found and when drivers are registered. It also identifies when input devices that use USB are found—like a mouse, keyboard, webcam, etc.

- c) Enter **dmesg -H | grep btusb**
- d) Verify that the kernel is reporting that a new interface driver was registered for the **btusb** module you installed earlier.

```
[student01@localhost modprobe.d]$ dmesg -H | grep btusb
[ +0.001415] usbcore: registered new interface driver btusb
[ +0.000052] ath9k_hw ath mac80211 iTCO_wdt iTCO_vendor_sup
uvcvideo snd_hda_codec_realtek videobuf2_vmalloc videobuf2_m
f2_core videodev snd_hda_codec_hdmi snd_hda_codec_generic i2c
```

Summary

In this lesson, you explored the purpose of the Linux kernel and leveraged its modular nature. You also monitored the kernel's operation to ensure it behaved as expected. By managing the kernel, you can customize Linux at a low level to ensure it suits your requirements.

Do you expect that you will need to load any special modules into the kernel to fulfill your organization's requirements? If so, what modules might you need to load?

Do you expect that you will need to configure any kernel parameters at runtime? If so, what kind of parameters might you configure?



Practice Question: Additional practice questions are available on course website.

Lesson 7

Managing the Linux Boot Process

LESSON TIME: 1 HOUR, 30 MINUTES

LESSON INTRODUCTION

Now that you've configured the kernel, you can learn more about how the kernel is loaded into memory and how the operating system actually starts. You'll also configure this boot process to your liking, ensuring that the Linux® workspace operates as intended from the very beginning.

LESSON OBJECTIVES

In this lesson, you will:

- Configure components that make up the Linux boot process.
- Configure the GNU GRUB 2 boot loader.

Topic A

Configure Linux Boot Components



EXAM OBJECTIVES COVERED

1.1 Explain Linux boot process concepts.

1.4 Given a scenario, manage storage in a Linux environment.

To begin with, you must become familiar with how exactly Linux boots, as well as identify various components that make up the boot process. In addition, you'll configure some of these components to alter the boot process to fit your needs.

BOOTING

Booting is the process of starting or restarting a computer and loading an operating system for the user to access. In the boot process, a booting environment reads a small program that is stored in read-only memory (ROM). This program then executes various operations in RAM that *bootstrap* the operating system and make it available for use.

Linux, like other operating systems, must be booted for it to function. There are various options associated with the boot process that you can configure, if necessary.

BOOT LOADER

A **boot loader** is the small program stored in ROM that loads the kernel from a storage device, and then starts the operating system. A boot environment like BIOS reads the boot loader from ROM so that the boot loader can execute the necessary operations to start the process.

Boot loaders are able to protect the boot process with a password to prevent unauthorized booting of the system. In addition, boot loaders can load more than one operating system into the computer's memory, but the user needs to select the desired operating system to use during boot.

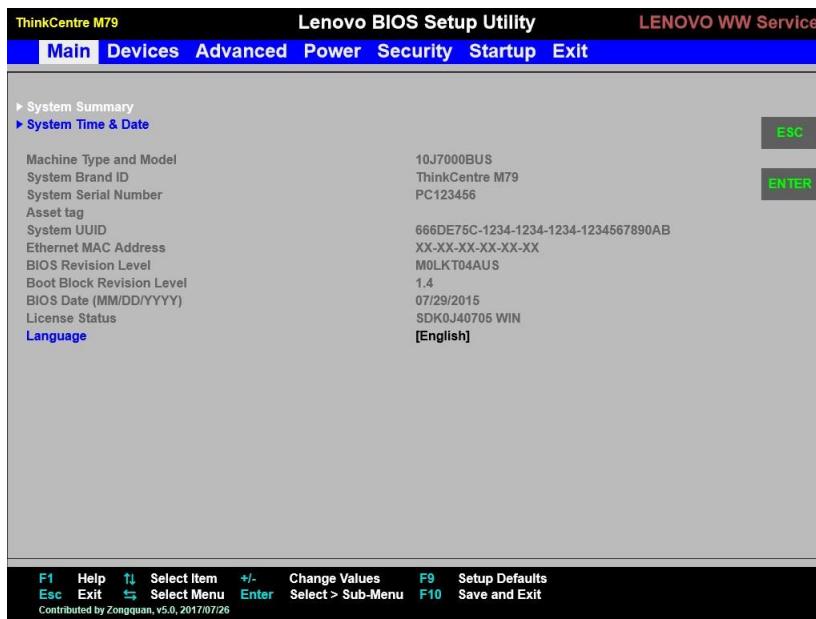
BOOT LOADER COMPONENTS

The boot loader uses three main components that work together to systematically load the operating system in stages.

Component	Description
Boot sector program	The first component of the boot loader. It is loaded by a boot environment on startup and has a fixed size of 512 bytes. Its main function is to load the second stage boot loader; however, it can also load another sector or a kernel.
Second stage boot loader	Loads the operating system and contains a kernel loader.
Boot loader installer	Controls the installation of drive sectors and can be run only when booting from a drive. It coordinates the activities of the boot sector and the boot loader.

BIOS

The **Basic Input/Output System (BIOS)** is a standard for firmware interfaces and is stored on a computer motherboard's ROM chip. When a computer with BIOS is powered on, the BIOS firmware is the first to run; this enables it to test the various hardware components in a computer, as well as run a boot loader so that an operating system can start. The BIOS has access to the ports used by basic hardware input devices like a mouse and keyboard. Users can also load up a BIOS interface instead of an operating system to make various hardware-level changes. For several decades, BIOS was the dominant standard in the home and enterprise computer industry.



A BIOS interface.

UEFI

Unified Extensible Firmware Interface (UEFI) is newer firmware technology that has largely replaced BIOS by bringing with it several key advantages. UEFI runs faster than BIOS, can operate within a greater amount of memory, can access storage drives of currently unattainable sizes, can access more hardware types, and has improved security protections. Most modern motherboards, as well as the pre-assembled PCs that use them, ship with UEFI.

Like BIOS, UEFI provides an environment with which to execute a boot loader, and ultimately start up the operating system for the user to work with.

PASSWORD PROTECTION

One security feature that both BIOS and UEFI include is the ability to set a password. If this password is not provided at boot time, the system will not boot. Since BIOS/UEFI firmware differs between hardware manufacturer, the process of setting this password is not consistent. However, most firmware places this password protection option in a "Security" or "Password" section.

ADDITIONAL BOOT OPTIONS

BIOS and UEFI are not the only environments you can boot Linux from. The following table describes some additional boot options.

Boot Option	Description
Boot from ISO	An ISO image is a system image, originally that of an optical disc. Today, it is commonly used as a file format for packaging and distributing images of operating systems that users can boot from, as well as use to install the OS. Typically, you'd write the ISO image to an optical disc or USB thumb drive, then insert the media into the computer and instruct a boot environment like UEFI to boot from that media. ISOs are also commonly used to construct virtual machines.
PXE	Preboot Execution Environment (PXE) is a part of the UEFI standard that enables a client to retrieve the necessary boot loader and system files from a server over the network. The client configures UEFI to boot from PXE, and during the startup process, it will search for Dynamic Host Configuration Protocol (DHCP) servers that also act as PXE servers. Once the proper server is found, the server transfers the necessary boot files to the client over the Trivial File Transfer Protocol (TFTP).
Boot from HTTP/FTP	Clients can also acquire boot data over a network from content delivery protocols like Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP). These are typically faster, more reliable, and more secure than the standard TFTP protocol used in PXE. Open source implementations of PXE, like iPXE, extend PXE support to include these protocols.
Boot from NFS	This is another network boot option. Rather than store system files on a local storage drive, a client will mount an NFS share as its root file system. The share must be prepared ahead of time and stored on an NFS server that the client can retrieve the files from. Therefore, the client does not store data locally, but on the NFS server. DHCP, TFTP, and other network protocols can be used to communicate the necessary boot data in such an environment.

SECTORS

A **sector** is the smallest unit of storage read from or written to a drive. A sector stores 512 bytes of data by default. On hard disk drives, a collection of sectors is called a track. The number of sectors in a track may vary, and so does their capacity to hold data. The size of a sector can be altered when formatting the drive.

```

root@server01:~#
Command (m for help): n
Partition number (7-128, default 7):
First sector (34-976773134, default 256147456):
Last sector, +sectors or +size{K,M,G,T,P} (2561474
56-976773134, default 976773134): +2G
Created partition 7

Command (m for help):

```

Working with sectors while partitioning a drive.

MBR

The **master boot record (MBR)** is the first physical sector on a storage drive and a type of partition structure. The MBR boot sector contains the boot loader that loads the operating system into memory. It also contains the partition table of the storage drive. MBR determines what sectors are available to each partition, as well as which partition is considered bootable and which partitions are not.

For many years, MBR was the dominant partition structure used in Linux and other operating systems. However, it has three major disadvantages:

- The maximum storage space of an MBR-partitioned drive is two terabytes.
- MBR-partitioned drives can have a maximum of four primary partitions.
- The boot data is stored in one sector, which increases the risk of corruption.

GPT

The **GUID Partition Table (GPT)** is a successor to MBR that makes up for the latter's shortcomings. Like MBR, it is a partition structure, but it employs a more modern design and is part of the UEFI standard. Every partition on a drive is assigned a globally unique identifier—a GUID—to distinguish it from every other partition on (theoretically) every drive.

The storage space and partition number maximums are so large that they are not currently achievable, and any limitations are going to be imposed by the file system type or operating system kernel, rather than GPT itself. GPT also has the advantage of storing its boot data in multiple locations on a drive to enhance redundancy. If the primary location is corrupted, GPT can leverage one of the other copies to restore the boot data.

Whenever possible, partitioning a drive with GPT is preferable to MBR.

RAW PARTITION

Other than formatting a partition as MBR or GPT, you can also format a partition as raw. A **raw partition** enables users and applications to read from and write to a block storage device directly, without using the system cache. This is useful in situations where software like a database management system (DBMS) has its own caching mechanism. The DBMS has greater control over I/O caching in a raw partition and can bypass the caching normally done by the kernel.

initrd

The **initial ramdisk (initrd)** refers to the root file system that is temporarily loaded into memory upon system boot. The initrd loads along with the kernel, which controls its functionality. The initrd enables the system to be started in two phases. In the first phase, the system is booted with the minimal set of modules required to load the main or the permanent root file system. In the second phase, when the main root file system is mounted, the previously mounted initrd file system is removed and the user space boot process continues.

The initrd is useful because there are many potential variables that can complicate the boot process. For example, the kernel needs to find and load the necessary device driver modules, as well as the actual root file system itself. There's also the possibility that the root file system uses one of several advanced storage methods, like LVM or NFS, which have different mount requirements than a standard partition. Rather than hardcode all of this behavior in the kernel and introduce bloat, the initrd's temporary root file system can handle these tasks.

THE initrd IMAGE

The Linux **initrd image** is an archive file containing all the essential files that are required for booting the operating system. It can be built or customized to include additional modules, remove unnecessary modules, or update existing modules.

Typically, this image is stored in the `/boot` directory.

THE mkinitrd COMMAND

The **mkinitrd** command is used to create the initrd image for preloading the kernel modules.

```
root@server01:~  
*** Including modules done ***  
*** Installing kernel module dependencies and firmware ***  
*** Installing kernel module dependencies and firmware done ***  
*** Resolving executable dependencies ***  
*** Resolving executable dependencies done***  
*** Hardlinking files ***  
*** Hardlinking files done ***  
*** Stripping files ***  
*** Stripping files done ***  
*** Generating early-microcode cpio image contents ***  
*** Constructing GenuineIntel.bin ****  
*** Store current command line parameters ***  
*** Creating image file ***  
*** Creating microcode section ***  
*** Created microcode section ***  
*** Creating image file done ***  
*** Creating initramfs image file '/boot/new-initrd.img' done ***
```

Creating a new initrd image.

Various options of the **mkinitrd** command are given in the following table.

Option	Used To
<code>--preload={module name}</code>	Load a module in the initrd image before the loading of other modules.

Option	Used To
--with={module name}	Load a module in the initrd image after the loading of other modules.
-f	Overwrite an existing initrd image file.
--nocompress	Disable the compression of the initrd image.

SYNTAX

The syntax of the **mkinitrd** command is **mkinitrd [options] {initrd image name} {kernel version}**

The following example creates an initrd image from the current kernel version and names the image **initrd-<kernel version>.img**:

```
mkinitrd /boot/initrd-$(uname -r).img $(uname -r)
```

THE /boot/ DIRECTORY

As defined by the Filesystem Hierarchy Standard (FHS), the **/boot/** directory contains files that are used to facilitate the Linux boot process. The following table describes some of the files and subdirectories in **/boot/** that are of note.

File or	Description
/boot/grub/	This directory contains configuration files for a type of boot loader called GRUB. The /boot/grub2/ directory does likewise, but for GRUB 2, an improved version.
/boot/efi/	This directory contains boot files for an EFI system partition (ESP), which is a required partition for systems that boot from UEFI. It contains boot loader, device driver, and system application files that are executed by UEFI. Boot loader files are typically named with a .efi extension.
/boot/initramfs-<kernel version>.img	This file is an initramfs image, which is an alternative to initrd that uses different methods to do the same basic thing: initialize a temporary root file system on boot. Whereas initrd requires a special driver to be compiled into the kernel, initramfs does not. In addition, the initrd image is a block device formatted with a fixed-size file system, whereas the initramfs image is an archive file that can be sized dynamically.
/boot/vmlinuz-<kernel version>	This is a compressed executable file that contains the Linux kernel itself. The boot loader loads this file into memory during the boot process to initialize the operating system. A related file is vmlinux , which is essentially the non-compressed version of the kernel used for debugging.

THE dracut COMMAND

The **dracut** command is used to generate an initramfs image, similar to how **mkinitrd** is used to generate an initrd image. In fact, on some distributions, **mkinitrd** is a compatibility wrapper that calls the **dracut** command.

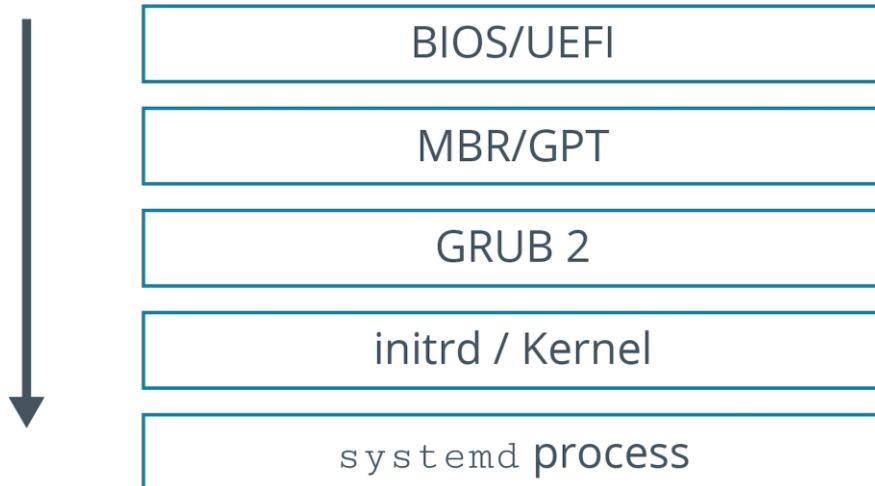
An example of using the **dracut** command to create an initramfs image is as follows:

```
dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

THE BOOT PROCESS

The boot process is repeated each time your computer is started by loading the operating system from a storage device. It involves a series of sequential steps that can be divided into BIOS/UEFI initialization, boot loader, kernel and initrd/initramfs initialization, and boot scripts.

The following is an example boot process that uses an initrd image:



A high-level look at the components involved in the Linux boot process.

1. The processor checks for the BIOS/UEFI firmware and executes it. This is also where the power-on self-test (POST) occurs.
2. BIOS/UEFI checks for bootable media from internal storage devices or peripherals like USB thumb drives and DVD-ROMs. It locates a valid device to boot the system.
3. BIOS/UEFI loads the primary boot loader from the MBR/GPT partition into memory. It also loads the partition table along with it.
4. The user is prompted by GRUB 2 to select the operating system they want to boot. If the user does not respond, then the default operating system will be booted.
5. The boot loader determines the kernel and locates the corresponding kernel binary. It then uploads the respective initrd image into memory and transfers control of the boot process to the kernel.
6. The kernel configures the available hardware drivers, including processors, I/O subsystems, and storage devices. It decompresses the initrd image and mounts it to load the necessary drivers. If the system implemented any virtual devices, such as LVM or software RAID, then they are initialized.
7. The kernel mounts the main root partition and releases unused memory. To set up the user environment, the `systemd` program is run. It becomes process ID 1.
8. The `systemd` program searches for the `default.target` file, which contains details about the services to be started. It mounts the file system based on the `/etc/fstab` file and begins the process of starting services. On most systems, the target will either be `multi-user.target` or `graphical.target`.
9. If graphical mode is selected, then a display manager like XDM or KDM is started and the login window is displayed on the screen.
10. The user enters a user name and password to log in to the system.

11. The system authenticates the user. If the user is valid, then various profile files are executed.
12. The shell is started and the system is ready for the user to work on.

KERNEL PANIC

Kernel panic is a mechanism by which the system detects there has been a fatal error and responds to it. A fatal error typically results in the system becoming unstable or totally unusable. Software that handles kernel panics will display an error message to the user and dump the current state of kernel memory to a storage device for later debugging. Depending on how the system is configured, the panic handler will either reboot the system automatically, or wait for the user to do so.

In Linux, kernel panic can happen for a number of reasons and at any point during operation, but it is usually experienced during the boot process. Common causes include the following:

- The kernel itself is corrupted or otherwise improperly configured.
- The **Systemd** program is not executed during boot, leaving the system unusable.
- The kernel cannot find or otherwise cannot mount the main root file system.
- Malfunctioning or incompatible hardware is loaded into the kernel on boot.



Note: The equivalent of a kernel panic in Microsoft Windows is the well-known Blue Screen of Death (BSOD).



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 7-1

Discussing the Linux Boot Process

SCENARIO

Answer the following questions to check your understanding of the topic.

1. Your coworkers are trying to decide between using UEFI or legacy BIOS for some of your systems. They ask you for your opinion. What are the advantages of UEFI that you can tell your coworkers?
 2. MBR is to BIOS as _____ is to UEFI.
 3. What is the importance of the initrd phase of the boot process?
 4. You've taken a new position as a Linux administrator and you discover that every Linux system contains a /boot/efi/ directory. What is the significance of this /boot/efi/ directory?
 5. Linux administrators don't enjoy seeing a kernel panic during the boot process because it means that the system won't recover on its own and something serious has happened. What are some possible causes of a kernel panic that occur during the boot process?

Activity 7-2

Identifying Linux Boot Components

SCENARIO

You need to get acquainted with the boot sequence components because understanding how they work will help you troubleshoot any issues that may arise when loading the operating systems onto new machines.

- 1. What must happen before the boot loader can run?**
 - 2. When the boot loader completes its tasks, what happens next?**
 - 3. What is the role of the MBR?**
 - 4. Which of the following are advantages that GPT has over MBR? (Choose two.)**
 - GPT has larger maximum storage space for partitions.
 - GPT enables you to select multiple operating systems to boot from.
 - GPT stores boot data in multiple locations for redundancy.
 - GPT can support multiple file system types.
 - 5. What is the purpose of systemd during the boot process?**
 - 6. Which boot process component configures device drivers?**
 - The boot loader.
 - The kernel.
 - The inittab file.
 - The BIOS/UEFI.

7. Which of the following boot options enables users to mount a file share as the root file system, rather than storing that file system locally?

- Boot from NFS
 - Boot from ISO
 - Preboot Execution Environment (PXE)
 - Boot from HTTP/FTP
-

Activity 7-3

Creating an initrd Image

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

As part of your server infrastructure, you plan on having some systems boot from an NFS share. The kernel in the deployed systems doesn't have an NFS module. Without this, your systems cannot mount an NFS share as the root file system. So, you need to create a new initrd image so that the kernel can successfully mount the share. First, however, you'll establish a baseline image that other images can build off of.

1. Create a new initrd image.
 - a) Enter **uname -r** to identify the current kernel.
 - b) Enter **sudo mkinitrd -v /boot/initrd-\$(uname -r).img \$(uname -r)**
\$(uname -r) substitutes the name of the kernel in this command.
 - c) Examine the verbose output from **mkinitrd** noting the various kernel modules that are included in the initrd image by default.
 - d) Enter **ls -l /boot** and verify that your new initrd image was created.
The image should be named **initrd-<kernel version>.img** and should have been last modified on today's date.
2. Create an initrd image with an NFS module installed.
 - a) Enter **sudo mkinitrd -v --with=nfsv4 /boot/initrd-\$(uname -r)-nfs.img \$(uname -r)**
 - b) Enter **ls -l /boot** and verify your new NFS image was created.
 - c) Examine the file sizes for both initrd images (the base image and the NFS image) and verify that the NFS image is larger.
This suggests that the additional NFS module was loaded into the image, as intended.

Topic B

Configure GRUB 2



EXAM OBJECTIVES COVERED

1.1 Explain Linux boot process concepts.

Now that you're familiar with the purpose and role of a boot loader in the boot process, you can focus on configuring the primary Linux boot loader: GRUB 2.

GNU GRUB

The **GNU GRand Unified Bootloader (GNU GRUB)** is a boot loader developed by the GNU Project that became popular on Unix-like systems. It enables users to choose which operating system or kernel version to boot in a multi-platform environment. Although the original version of GRUB was the primary boot loader for Linux distributions, it had several limitations and was eventually phased out in favor of a newer version of GRUB. This original version is sometimes referred to as GRUB legacy.

GRUB 2 IMPROVEMENTS

GRUB 2 is more than simply a newer version of GRUB; it is a complete redesign and rewrite of the GRUB system. GRUB 2 offers administrators more control over the boot process, boot devices, and boot behavior. In addition, it comes with several improvements, including:

- Support for non-x86 architecture platforms.
- Support for live booting (booting an OS from storage media and running the OS entirely in memory, without installation).
- Support for partition UUIDs.
- Support for dynamically loading modules that extend GRUB's functionality.
- Ability to configure the boot loader through scripts.
- Rescue mode, which attempts to fix boot issues like corrupted or missing configurations.
- Support for custom graphical boot menus and themes.

Because of these improvements, GRUB 2 has become the default boot loader on almost all modern Linux distributions.

GRUB 2 INSTALLATION

The **grub2-install** command is used to install the GRUB 2 boot loader on a storage device. It copies GRUB 2 files into the `/boot/grub2` directory and, on some platforms, installs GRUB 2 into the boot sector. However, **grub2-install** applies to BIOS systems, not UEFI. To install GRUB 2 on a UEFI system, use a package manager to install the **grub2-efi** package. Installing this package will copy GRUB 2 files onto the EFI system partition (ESP) in the `/boot/efi` directory.

SYNTAX

The syntax of the `grub2-install` command is `grub2-install [options] [device name]`

grub2-install COMMAND OPTIONS

The following are some options you can use with the `grub2-install` command:

Option	Used To
<code>--modules {module names}</code>	Preload the specified kernel modules with the GRUB 2 boot loader.
<code>--install-modules {module names}</code>	Install only the specified modules and their dependencies, rather than the default of installing all available modules.
<code>--directory {directory name}</code>	Install files from the specified directory, rather than the default.
<code>--target {target platform}</code>	Specify the target platform to install GRUB 2 for, rather than the platform that is currently running.
<code>--boot-directory {directory name}</code>	Specify the boot directory to install GRUB 2 files to, rather than the default <code>/boot/</code> directory.
<code>--force</code>	Install GRUB 2 regardless of detected issues.

THE grub.cfg FILE

The `grub.cfg` file is the main configuration file for the GRUB 2 boot loader. On BIOS systems, it is located in the `/boot/grub2/` directory. On UEFI systems, it is located in the `/boot/efi/EFI/<distro>/` directory. For example, on CentOS 7, the path is: `/boot/efi/EFI/centos/grub.cfg`

```
root@server01:~#
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub2-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#
### BEGIN /etc/grub.d/00_header ###
set pager=1
if [ -s $prefix/grubenv ]; then
  load_env
fi
if [ "${next_entry}" ] ; then
  set default="${next_entry}"
  set next_entry=
  save_env next_entry
} } } }
```

Contents generated by command

The `grub.cfg` file.

This file is an executable shell script. Don't edit this file directly, as it is generated using a specific command that leverages configuration scripts stored elsewhere on the file system.



Note: On RHEL/CentOS 7, `/etc/grub2.cfg` is a symbolic link to the `grub.cfg` file. This maintains a legacy path option, as the original GRUB configuration file used to be located at `/etc/grub.cfg`.

THE `/etc/grub.d/` DIRECTORY

The `/etc/grub.d/` directory contains scripts that are used to build the main `grub.cfg` file. Each script provides various functions to GRUB 2 and is numbered so that the scripts can execute in a sequence. It's usually not a good idea to edit the existing scripts in this directory. If you want to add a custom script, then you can place it in this directory with a `##_` file name prefix, depending on what order you want the script to be executed in. You can also add your script to the existing `40_custom` file so that it executes last by default.

GRUB 2 BOOT MENU CUSTOMIZATION

The `/etc/grub.d/40_custom` file enables the customization of the menu presented to the user during the boot process. GRUB 2 will offer the user a menu of installed operating systems to choose from. This choice is useful for multi-boot scenarios (more than one operating system available on the computer), booting to different Linux kernels, or for booting into a rescue mode. The menu contents may be customized by editing the `/etc/grub.d/40_custom` file, enabling an administrator to specify the order of the menu choices, provide user-friendly names, and to password protect menu entries.

```
root@server01:~  
#!/bin/sh  
exec tail -n +3 $0  
# This file provides an easy way to add custom menu entries  
. Simply type the  
# menu entries you want to add after this comment. Be care  
ful not to change  
# the 'exec tail' line above.  
set superusers="student01"  
password_pbkdf2 student01 grub.pbkdf2.sha512.10000.794B908F  
604C8CA6DC75EBD35373A4A33B55BB3982C808C7FFAA6DD460419A9AD6F  
97A56E996B848F09A451B71AC3504799FAFE5F7D4D112AD61D60C4E713F  
11.31AAE31C9AAA19EB616C0E75E5C55B3BA49022B66E3490683ED807BC  
E0647EFE0BE10A5753AABCF3F3BAD8C3F3F97BB32FEC0BC5C809BFCB1  
9309AB601F1C6
```

Password protection config

Protecting the GRUB 2 boot menu with a password.

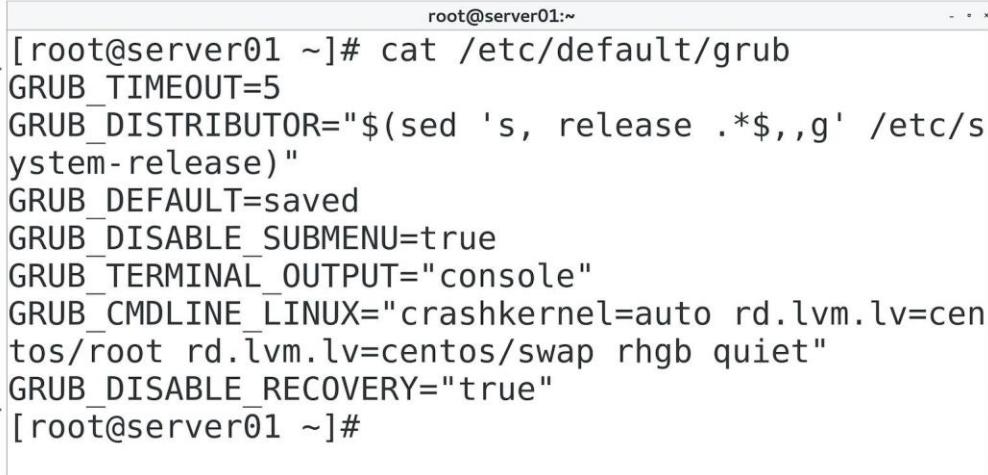
GRUB 2 PASSWORD GENERATION

You can generate a password hash to protect the boot menu by using the `grub2-mkpasswd-pbkdf2` command.

THE `/etc/default/grub` FILE

The `/etc/default/grub` file contains GRUB 2 display menu settings that are read by the `/etc/grub.d/` scripts and built into the `grub.cfg` file. It enables

you to change options such as how many seconds GRUB 2 will wait before automatically selecting the default boot option; whether or not GRUB 2 will order kernel versions in a sub-menu; whether or not GRUB 2 will display the menu in a graphical terminal; and more.



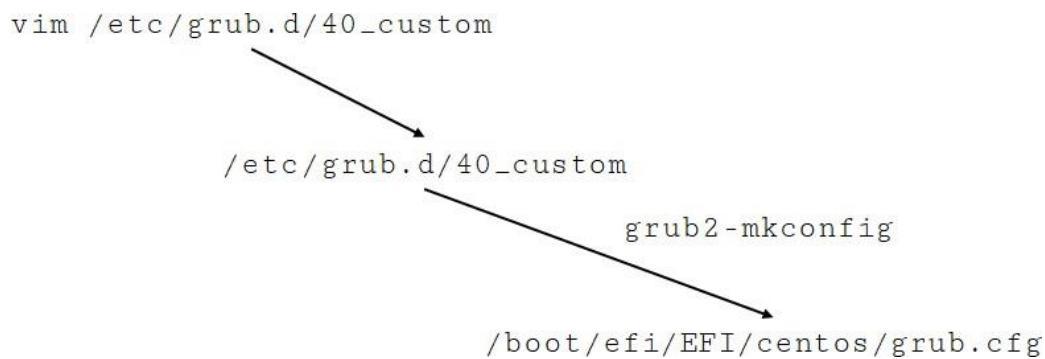
The screenshot shows a terminal window titled "root@server01:~". The command "cat /etc/default/grub" is run, displaying the following configuration settings:

```
root@server01 ~]# cat /etc/default/grub
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=$(sed 's, release .*$,,g' /etc/system-release)
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
[root@server01 ~]#
```

The /etc/default/grub file.

THE grub2-mkconfig COMMAND

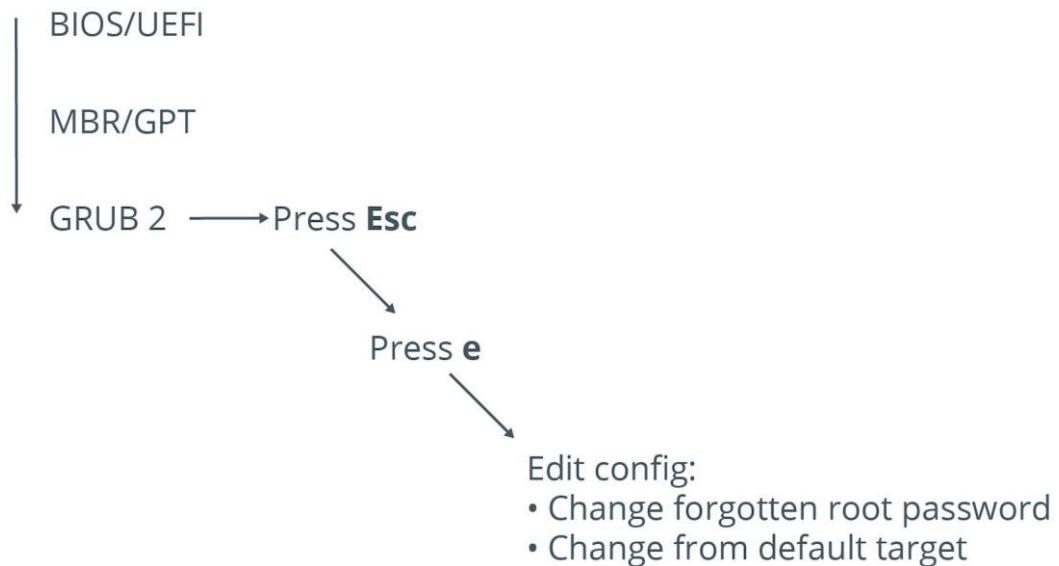
The **grub2-mkconfig** command generates a new **grub.cfg** configuration file, and is used to update an existing **grub.cfg** file. The **grub2-mkconfig** command combines the configuration file templates in the **/etc/grub.d/** directory with the settings in **/etc/default/grub** to generate the **grub.cfg** configuration file.



The process of editing GRUB 2 configuration files.

Note: On some distributions, this command is simply **grub-mkconfig**





The process of configuring GRUB 2 from the boot menu.

SYNTAX

The syntax of the `grub2-mkconfig` command is `grub2-mkconfig [-o {file name}]`



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 7-4

Discussing GRUB 2

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You want to create a bootable DVD-ROM Linux distribution for some kiosk computers that customers will use to look up products while in your retail locations. Which new GRUB 2 improvement makes this an easier task compared to a few years ago?

2. You decided to make all your systems more secure by adding a GRUB 2 boot loader password. After making the change to multiple systems, you feel confident that the new security measure will be effective in securing your systems. Which command can you use to generate a password for GRUB 2?

3. You have tasked a junior administrator with adding a single boot loader option to three of your Linux systems. The administrator lets you know that he was unsuccessful at making the changes because he received a permission denied error. What is the issue?

4. How can you edit the GRUB 2 boot loader during the boot process?

5. What are some examples of changes you could make to the default GRUB 2 menu?

Activity 7-5

Configuring GRUB 2

BEFORE YOU BEGIN

You are logged in to the CLI as your student account.

SCENARIO

Some of your fellow administrators are claiming that their Linux servers aren't booting properly. You are assigned to the task of troubleshooting these issues. You find that someone has modified the settings in the boot loader because there is no password protection. After correcting the boot configuration, you decide to protect GRUB 2 with a password so that only authorized users can modify it.

1. Verify that GRUB 2 is installed.

- a) Enter **sudo ls -l /boot/efi/EFI/centos**
- b) Verify that **grub.cfg** exists in this directory.

The presence of this file usually indicates that GRUB 2 is successfully installed on the EFI system partition.

- c) Enter **sudo cat /boot/efi/EFI/centos/grub.cfg** and verify that the configuration file is populated.

```
[student01@localhost modprobe.d]$ sudo cat /boot/efi/EFI/centos/grub.cfg
#
# DO NOT EDIT THIS FILE
#
# It is automatically generated by grub2-mkconfig using templates
# from /etc/grub.d and settings from /etc/default/grub
#
### BEGIN /etc/grub.d/00_header ###
set pager=1

if [ -s $prefix/grubenv ]; then
    load_env
fi
```

2. Create a password to lock the GRUB 2 configuration with.

- a) Enter **sudo grub2-mkpasswd-pbkdf2 | sudo tee -a /etc/grub.d/40_custom**

You're redirecting the output to the custom configuration file. You'll clean up this file shortly.

- b) Enter **Pa22w0rd** as the password.
- c) Reenter the same password.
- d) Verify that the PBKDF2 password is generated.

PBKDF2 uses a cryptographic technique called hashing to protect the password in storage.

3. Adjust the custom GRUB 2 configuration file to require your password.
 - a) Using **Sudo**, open **/etc/grub.d/40_custom** in the text editor of your choice.
 - b) Move the cursor to the line that says "Enter password:" and cut this entire line.
 - c) Cut the line after it that shows the reenter password prompt.
 - d) From the "PBKDF2" line, delete the string of text that says "PBKDF2 hash of your password is".
 - e) On the same line, replace the text you deleted with **password_pbkdf2 student##** where **##** refers to your student number.
 - f) Insert a new line above this that says:
set superusers="student##"

```
set superusers="student01"
password_pbkdf2 student01 grub.pbkdf2.sha512.10000.794B908F
604C8CA6DC75EBD35373A4A33B55BB3982C808C7FFAA6DD460419A9AD6F
97A56E996B848F09A451B71AC3504799FAFE5F7D4D112AD61D60C4E713F
11.31AAE31C9AAA19EB616C0E75E5C55B3BA49022B66E3490683ED807BC
E0647EFE0BE10A5753AABCD3F3BAD8C3F3F97BB32FEC0BC5C809BFCFB1
9309AB601F1C6
```

- g) Save and close the file.
4. Update the main GRUB 2 configuration file to apply your changes.
 - a) Enter **Sudo grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg**
 - b) Verify that no errors are returned and that the "done" message is displayed.
 This indicates that the new GRUB 2 configuration file has been generated successfully.

5. Test the password from the GRUB 2 boot menu.
 - a) Enter **reboot**
 - b) On the GRUB 2 boot menu screen, press **Esc** to stop the default selection timer.
 - c) Press **e** to edit the GRUB 2 configuration.
 - d) At the **Enter username** prompt, enter your student account name.



Caution: Input your user name and password very carefully, as you will be unable to edit any mistakes.

- e) At the **Enter password** prompt, enter **Pa22w0rd**
- f) Verify that you can see the GRUB 2 configuration on the screen.
- g) Press **Esc** to exit editing mode.
- h) Press **Enter** to boot back into the default selection.
- i) Log back in as your student account.

Summary

In this lesson, you identified various components that are essential to the Linux boot process, as well configured the GRUB 2 boot loader to your desired settings. By doing this, you can gain greater control over how Linux boots and how users can leverage boot options.

What boot options do you use or plan to use in your Linux environments? Why?

Do you anticipate the need to configure GRUB 2 in your environments? Why or why not?



Practice Question: Additional practice questions are available on the course website.

Lesson 8

Managing System Components

LESSON TIME: 3 HOURS

LESSON INTRODUCTION

So far, you've managed several elements of Linux® that are fundamental to its operation. This will help you get your systems up-and-running. You're not done, however; there are still specific components and features that you can fine-tune to meet your needs. In this lesson, you'll start by managing components that contribute to the overall system environment.

LESSON OBJECTIVES

In this lesson, you will:

- Configure localization options such as character sets and environment variables.
- Configure graphical user interfaces (GUIs).
- Manage services.
- Troubleshoot process issues.
- Troubleshoot system performance issues by analyzing the CPU and memory.

Topic A

Configure Localization Options



EXAM OBJECTIVES COVERED

1.6 Given a scenario, configure localization options.

Before users can get comfortable working in the Linux environment, they may need to have the system localized to their specific region, culture, or preferences. In this topic, you'll configure localization options for users who need them.

LOCALIZATION

In the world of operating systems, **localization** is the process of adapting system components for use within a distinct culture, other than the culture that the system was originally designed for. In a practical sense, this usually means translating interface components into specific languages; converting measurements into the system used in a specific region; configuring the keyboard layout that the user works with; setting the date and time attributes of a specific location; and more.

Localizing a Linux system is important for organizations that provide Linux services to personnel and customers all over the world. An administrator in Japan will likely be more comfortable working on a Linux server if that server is localized to use the Japanese language, rather than English. Working with the local time and with a compatible keyboard will also enhance the user experience for international users.

THE /usr/share/zoneinfo/ DIRECTORY

The `/usr/share/zoneinfo/` directory is a container for all of the regional time zones that you can configure the system to use. Subdirectories in this container usually organize languages by region; for example, the **Africa** subdirectory includes time zone files for specific countries or cities within the continent.

The individual files are not raw text files, but are special files used by the system. One way to change the system's time zone is by creating a symbolic link to one of these individual time zone files to the `/etc/localtime` file.

```
root@server01:~# ls -l /usr/share/zoneinfo/Africa
total 216
-rw-r--r--. 11 root root 156 Jan 30 2018 Abidjan
-rw-r--r--. 1 root root 828 Jan 30 2018 Accra
-rw-r--r--. 11 root root 271 Jan 30 2018 Addis Ababa
-rw-r--r--. 1 root root 734 Jan 30 2018 Algiers
-rw-r--r--. 11 root root 271 Jan 30 2018 Asmara
-rw-r--r--. 11 root root 271 Jan 30 2018 Asmera
-rw-r--r--. 11 root root 156 Jan 30 2018 Bamako
-rw-r--r--. 10 root root 157 Jan 30 2018 Bangui
-rw-r--r--. 11 root root 156 Jan 30 2018 Banjul
-rw-r--r--. 1 root root 194 Jan 30 2018 Bissau
-rw-r--r--. 8 root root 157 Jan 30 2018 Blantyre
-rw-r--r--. 10 root root 157 Jan 30 2018 Brazzaville
```

The `/usr/share/zoneinfo` directory.

THE `/etc/timezone` FILE

In some Debian-based distros, `/etc/timezone` can be used to view the time zone. This text file lists the time zone by the region structure you'd see in the `/usr/share/zoneinfo` directory. For example, the file might include the text `Europe/Berlin` to indicate that the system is using the zone that this city is in.

THE `date` COMMAND

The `date` command is used to print the date in a specified format. The `date` command will print the date based on the `/etc/localtime` file. By default, it will print the date in the following format:

```
<day of week> <month> <day> <24 hour time ##:##:#> <time zone>
<year>
Wed Oct 31 15:03:16 GMT 2018
```

You can also format the time using a number of different formatting options. You initialize the formatting options with a plus sign (+), and each option is prefaced with a percent sign (%). For example, to retrieve the week number (out of 52 weeks a year), you'd enter `date +%V`

You can also use the `date` command to change the system's date by including the `-s` option with a provided argument.

```
root@server01 ~]# date +%V
02
```

Week number

Retrieving the current week number.

SYNTAX

The syntax of the `date` command is `date [options] [format]`

FORMATTING OPTIONS

The following table lists some of the formatting options available.

Formatting	Prints
<code>%A</code>	The full weekday name.
<code>%B</code>	The full month name.
<code>%F</code>	The date in YYYY-MM-DD format.
<code>%H</code>	The hour in 24-hour format.
<code>%I</code>	The hour in 12-hour format.
<code>%j</code>	The day of the year.
<code>%S</code>	Seconds.
<code>%V</code>	The week of the year.
<code>%x</code>	The date representation based on the locale.
<code>%X</code>	The time representation based on the locale.
<code>%Y</code>	The year.

THE `timedatectl` COMMAND

The `timedatectl` command is used to set system date and time information. It can take one of several subcommands, as detailed in the following table.

Subcommand	Used To
<code>status</code>	Show the current date and time information, including local time, universal time, RTC time, time zone, and more. This is the same as issuing <code>timedatectl</code> by itself.
<code>set-time</code>	Set the system's time to the time provided. The format should be as follows: <code>2018-10-31 15:03:16</code>

Subcommand	Used To
set-timezone	Set the system's time zone to the time zone provided. The zone is in the format specified by the /usr/share/zoneinfo structure.
list-timezones	List all available time zones in the format specified by the /usr/share/zoneinfo structure.
set-ntp {0 1}	Enable or disable synchronization with a Network Time Protocol (NTP) server.

```
root@server01 ~]# timedatectl status
    Local time: Tue 2019-01-08 14:26:44 GMT
    Universal time: Tue 2019-01-08 14:26:44 UTC
        RTC time: Tue 2019-01-08 14:26:44
        Time zone: Europe/London (GMT, +0000)
    NTP enabled: yes
NTP synchronized: yes
RTC in local TZ: no
    DST active: no
Last DST change: DST ended at
                  Sun 2018-10-28 01:59:59 BST
                  Sun 2018-10-28 01:00:00 GMT
Next DST change: DST begins (the clock jumps one hour forward) at
                  Sun 2019-03-31 00:59:59 GMT
                  Sun 2019-03-31 02:00:00 BST
[root@server01 ~]#
```

Displaying time and date information.

SYNTA

X

The syntax of the **timedatectl** command is **timedatectl [options] [subcommand]**

timedatectl COMMAND OPTIONS

The following table lists some of the options for the **timedatectl** command.

Option	Used To
-H {remote host}	Execute the operation on the remote host specified by IP address or hostname.
--no-ask-password	Prevent the user from being asked to authenticate when performing a privileged task.
--adjust-system-clock	Synchronize the local (system) clock based on the hardware clock when setting the hardware clock.
-M {local container}	Execute the operation on a local container.

CLOCK TYPES

The **timedatectl** command exposes three different clocks that the system can use:

- The local clock. This clock reflects the current time in the system's locale (i.e., the time zone).
- The universal time. This clock reflects the time irrespective of the local time zone. It uses the international standard Coordinated Universal Time (UTC), which is the same as Greenwich Mean Time (GMT).

- The hardware clock. As the name implies, this clock functions at the hardware level and keeps time even when the computer is powered off. An OS will derive the current time from this hardware clock. Any OS can also adjust the hardware clock, but it's usually a good idea to keep it at UTC. The hardware clock is also known as the real-time clock (RTC).

SETTING THE HARDWARE CLOCK

Using **timedatectl** it is possible to set the hardware clock to be equal to the local time. However, this is not advisable because the RTC is not automatically updated. It is only updated by external facilities like other installed operating systems. Multiple OSs can adjust the RTC for daylight savings time (DST), causing an over-correction.

THE hwclock COMMAND

The **hwclock** command enables you to view and set the hardware clock. As mentioned before, it is strongly recommended that you keep the hardware clock aligned with UTC to prevent over-correction by other operating systems.

You can also use the **hwclock** command to adjust the systematic drift. The **systematic drift** is the predictable amount of time that the hardware clock gains or loses each day, making it inaccurate and throwing it out of alignment with the system clock. The **/etc/adjtime** file records information about when and by how much the hardware clock is changed. The **hwclock** command can then consult this file to identify the drift value, which it can use to correct the clock's time.

SYNTAX

The syntax of the **hwclock** command is **hwclock [options]**

hwclock COMMAND OPTIONS

The following are some of the command options used with the **hwclock** command.

Option	Used to
--set	Set the hardware clock to the provided date and time.
-u	Set the hardware clock to UTC.
-s	Set the system time from the hardware clock.
--adjust	Add or subtract time from the hardware clock to account for systematic drift.

THE localectl COMMAND

The **localectl** command is used to view and configure the system locale and keyboard layout settings. A system's locale determines how it will represent various culture-specific elements, the most prominent of which is the language used in the interface. However, a locale can also determine factors such as how date and time are formatted, how monetary values are formatted, and more. Keyboard layouts can be configured independently of the locale and will determine how each physical key press is interpreted by the operating system. There are many keyboards with different physical layouts, so the system needs to be configured with the correct one or else the wrong character may be entered.

Like the **timedatectl** command, the **localectl** command offers various subcommands for managing the system locale and keyboard layout.

Subcommand	Used To
status	Show the current locale and keyboard layout. This is the same as issuing localectl by itself.
set-locale	Set the system locale to the locale provided.
list-locales	List all available locales on the system.
set-keymap	Set the keyboard layout to the provided layout.
list-keymaps	List all available keyboard layouts on the system.

```
root@server01 ~]# localectl status
  System Locale: LANG=en_US.UTF-8
    VC Keymap: us
    X11 Layout: us
[ root@server01 ~]#
```

Displaying locale settings.

SYNTA

X

The syntax of the **localectl** command is **localectl [options] [subcommand]**

localectl COMMAND OPTIONS

The following table lists some of the options for the **localectl** command.

Option	Used To
-H {remote host}	Execute the operation on the remote host specified by IP address or hostname.
--no-ask-password	Prevent the user from being asked to authenticate when performing a privileged task.
--no-pager	Prevent the output from being piped into a paging utility.
--no-convert	Prevent a keymap change for the console from also being applied to the X display server, and vice versa.

CHARACTER SETS AND ENCODING

Character **encoding** is the process of converting text into bytes, and **decoding** is the process of converting bytes into text. Both of these concepts are important because text is much easier for humans to interact with, whereas computers process data in bytes. Therefore, there needs to be an intermediary process that enables both entities to interface with the same information.

In many systems, the default encoding is UTF-8 using the Unicode character set. For example, the capital letter C is associated with the positional number **U+0043** in Unicode. UTF-8 encodes this number (43) in binary as **01000011**. However, not all software uses this encoding. For example, some software might default to an older encoding standard like ASCII. If you've ever opened a text file and seen garbled and unreadable characters and symbols, then the text editor is probably assuming the wrong encoding.

In Linux, the locale settings determine what encoding scheme the system will use. The same general locale may have different encoding options. For example, **de_DE.utf8** and **de_DE.iso88591** both set Germany and the German language as the locale, but the former sets the encoding as UTF-8 and the latter sets an ASCII encoding standard.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 8-1

Discussing Localization Options

SCENARIO

Answer the following questions to check your understanding of the topic.

1. The date command is very powerful, especially for use in scripts. You can add timestamps to log files and to script output. How can you display the day, date, and time using the date command?
 2. You recently moved several Linux server systems from your local data center in the Eastern U.S. time zone to a disaster recovery data center in the Central U.S. time zone. You need to change time zones for those systems. What steps can you take to change the time zones?
 3. Your company has deployed physical systems in its data centers rather than virtual ones. Physical systems often experience some time drift if not calibrated with an external source, such as an NTP server. You should set the hardware clock to UTC and also correct any time differences by telling the hardware clock to compensate for drift. How can you set the hardware clock and adjust for drift?
 4. Your company is global in scope and you have systems in data centers in several different countries. Often while typing you discover that some odd characters appear on the screen. One of your counterparts in one of the offshore countries informs you that they have changed the keymap to better suit their needs. How can you change the keymap for U.S. compatibility?

5. **Why would you need to change the locale settings of a system based on its global location?**
-

Activity 8-2

Configuring Localization Options

BEFORE YOU BEGIN

You are signed in to the CLI as your student account.

SCENARIO

One of your colleagues is located remotely—in London, England. Just like you, he needs to be able to log in to Develetech's Linux servers in order to administrate them. The server he needs to remotely administrate is located in the US, even though it primarily services users in Great Britain. So, you'll set this server to use the time zone in London, as well as change the language and keyboard layout settings to those of Great Britain. This will make it easier for your English colleague to work within the environment and for the server to operate within the correct time zone.

1. Retrieve your system's current date and time information.
 - a) Enter **timedatectl**
 - b) Verify that you are given information such as:
 - The local time.
 - The universal time.
 - The real-time clock (RTC) time (i.e., hardware clock).
 - The time zone.
 - Network Time Protocol (NTP) information.
 - Daylight savings time (DST) information.

2. Find the time zone for London.
 - a) Enter **timedatectl list-timezones**
 - b) Verify that there are several pages of time zones available.

Time zones are categorized by global region, then typically by city or small nation. This information is pulled from the time zone files and directories in **/usr/share/zoneinfo**
 - c) Press **q** to quit and return to the prompt.
 - d) Enter **timedatectl list-timezones | grep London** to filter the results.

The relevant time zone is in the format **Europe/London**.

3. Change the time zone to London's current time zone.
 - a) Enter **sudo timedatectl set-timezone Europe/London**
 - b) Enter **timedatectl**
 - c) Verify that the time zone has changed and that the local time reflects this change.

The time zone that is set will depend on the time of year. From the last Sunday in March to the last Sunday in October, the time zone will be British Summer Time (BST). The rest of the year, London is on Greenwich Mean Time (GMT), otherwise referred to as UTC.

- d) Verify that the universal time is either the same as the local time or is one hour behind, depending on the time of year.
The universal time is synonymous with GMT/UTC and is used as a global reference point.
- e) Examine the RTC time.
This is the hardware clock, and it is set by the OS. Many Linux distros set this to UTC by default, including CentOS. You'll leave this as-is.
- f) Enter **date** to confirm the date and time on the system.

4. Find the appropriate locale settings for Great Britain.

- a) Enter **localectl status** and note your system's current language locale and keyboard layout.



Caution: Take note of these values if your locale and keyboard layout are not US. You will need to revert to these values at the end of the activity.

- b) Enter **localectl list-locales**
- c) Page through the list of available locales until you get to the locales that start with **en_** (English language).
- d) Find the **en_GB.utf8** locale.
This is a locale for Great Britain that uses UTF-8 encoding.
- e) Press **q** to quit.
- f) Enter **localectl list-keymaps**
- g) Verify that you can find a keyboard layout named **gb** indicating Great Britain.

5. Configure the appropriate locale settings for Great Britain.

- a) Enter **sudo localectl set-locale "LANG=en_GB.utf8"**
- b) Enter **sudo localectl set-keymap gb**
- c) Enter **localectl status** and verify that the language locale and keyboard layout are both set to Great Britain.
- d) On your keyboard, press the **Backslash** key and verify that it types a number symbol (#).
This is due to layout differences between US and GB/UK keyboards.



Note: This will, of course, only work if you're using a US keyboard.

6. For classroom purposes, revert the locale settings.

- a) Enter **sudo localectl set-locale "LANG=en_US.utf8"**



Note: Press the **Up Arrow** until you retrieve the command you used to set the locale to Great Britain. Then you can just replace the text.



Caution: If your original language and keyboard layout are something other than US, you will need to enter your original values instead.

- b) Enter **sudo localectl set-keymap us**
- c) Enter **localectl status** and verify that the original locale options are set.

Topic B

Configure GUIs



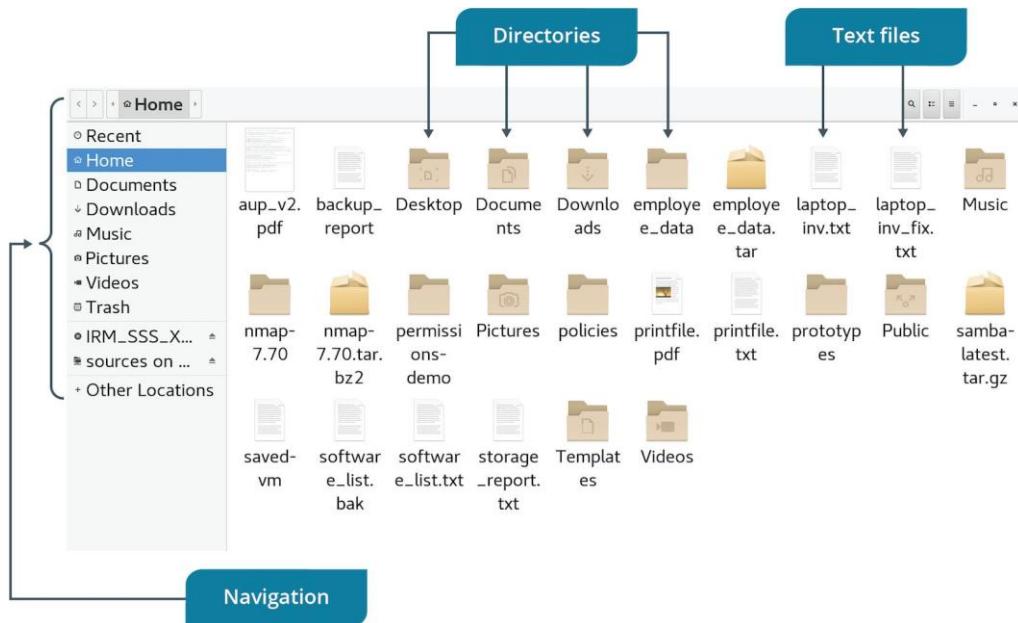
EXAM OBJECTIVES COVERED

2.8 Compare and contrast Linux graphical user interfaces.

The CLI is not the only interface that Linux is capable of providing to the user. In some cases, users will want to work with a graphical user interface (GUI) for a more visual experience. So, in this topic, you'll configure multiple GUIs to help certain users be more productive and access features that aren't available in a non-visual CLI.

GUIs

A **graphical user interface (GUI)** enables users to interact with a system or application through visual design elements rather than pure text as in a command-line interface (CLI). In Linux, a GUI provides similar functionality to the GUIs of other operating systems like Windows® and macOS®. Users can select icons that represent files, folders, and programs to open or modify them; configure settings through menus; interact with applications through a windowed interface; and work from a desktop environment.



A file browser GUI app.

As you've seen, GUIs in Linux are optional and are commonly used in workstation or other non-server roles. GUIs add an extra layer of performance overhead, especially since drawing visual elements consumes much more processing power and memory than simple text. So, they are less commonly used on servers where performance is paramount. Still, this is not universally true, as many administrators prefer to work in a GUI environment, or at least have access to one even if they spend most of their time

entering commands. GUIs can emulate a CLI that you enter commands into, so you're not limited to using the visual interface.

DISPLAY SERVERS

A **display server** is the component of a GUI that constructs and manages the windowing system and other visual elements that can be drawn on the screen. Display servers accept client input requests and send them to appropriate kernel modules for processing. Likewise, they also manage the process of receiving requests from an application to display output to the client.

As the name implies, display servers manage communications over a specific network-aware protocol. This enables remote clients to access GUI elements of a Linux system, like specific windows. However, the server can still provide its services to local clients.

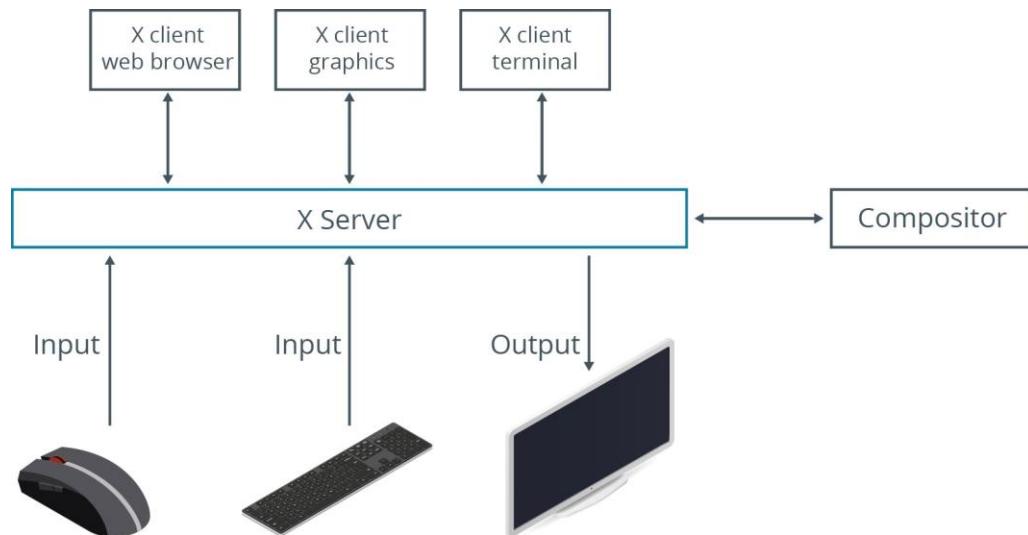
Linux supports several different display servers. Two of the most prominent are implemented in the X Window System and Wayland.

THE X WINDOW SYSTEM

The **X Window System**, also known as **X11** or **X**, is a platform-independent display server and windowing system that was developed by the Massachusetts Institute of Technology (MIT) in 1984.

Like all display servers, the X server coordinates client input and application output to determine how to draw elements on the screen. The X server also communicates with a separate compositor. The compositor reads a memory buffer that each application writes to, then uses the information in this buffer to combine each individual application window on the screen so that multiple windows can appear at once.

Whenever the X server receives an event (e.g., a button was clicked and must now be highlighted), it must inform the compositor so that it can re-compose the portion of the screen that is affected by the event.



The X Window System architecture.

X.ORG SERVER

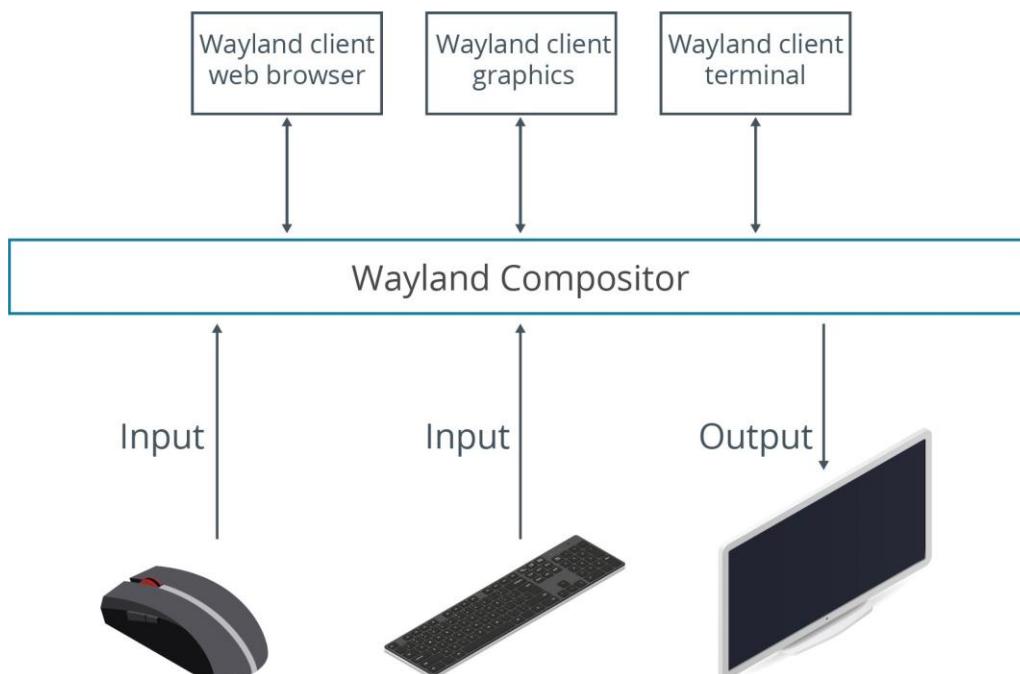
X.Org Server is the free and open source reference implementation of the X Window System for Linux and other Unix-like operating systems. Released in 2004, it quickly became the *de facto* display server on many Linux distributions. Prior to X.Org Server, the main implementation of X in Linux was XFree86, which eventually changed to a

custom license that the free software community found GPL-incompatible. Thus, XFree86 was forked into X.Org Server.

WAYLAND

Wayland is both a display server and its reference implementation in Unix-like operating systems that is meant to improve upon and replace the X Window System. The primary difference between Wayland and X is that, in Wayland, the compositor *is* the server rather than a separate component. This enables clients to exchange events directly with the compositor, cutting out the X server as a middle man.

Wayland was first released in 2008, and although X.Org Server still dominates in Linux distributions, adoption of Wayland has been slowly increasing. For example, Fedora® started using Wayland as its default display server starting with version 25, released in November of 2016.



The Wayland architecture.

X VS. WAYLAND

Wayland offers several improvements over X, including:

- In X, the X server must determine which window an event applies to. It can't always do this correctly, because the separate compositor controls how the window is redrawn through actions like resizing and rotation—information that the X server doesn't necessarily understand. Because the compositor and server are one in Wayland, this is not an issue.
- In X, the compositor must fetch event data from the server, which can introduce latency. In Wayland, the compositor receives events directly from the client, mitigating latency issues.
- Wayland simplifies the graphical rendering process by enabling the client to perform its own rendering.
- Although Wayland is not network-aware in the same way as X, it can still leverage remote desktop protocols for controlling a GUI environment over a network.

- Older implementations of X do not isolate the data I/O of each window, whereas Wayland does. This helps ensure the security of data.

DISADVANTAGES TO WAYLAND

Although Wayland improves upon X in almost every way, there are still some issues with Wayland. For example, Canonical® made Wayland the default display server for Ubuntu® 17.10 in October 2017. In April 2018, for the release of Ubuntu 18.04, Canonical announced that it was switching back to X.Org Server for three primary reasons:

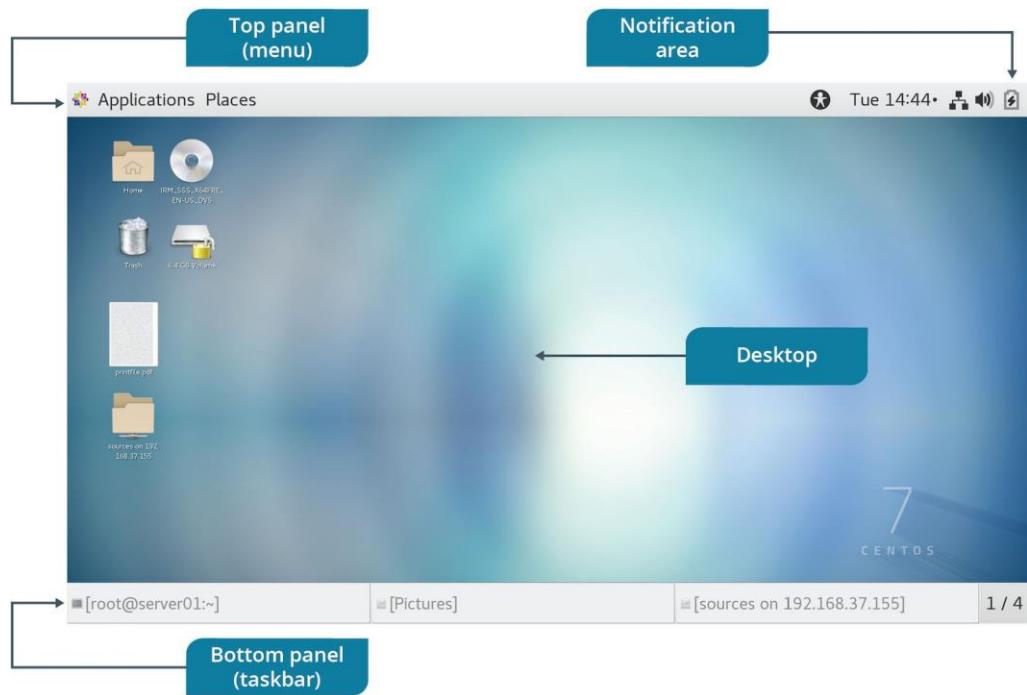
- Screen sharing software tends to work better under X.Org Server than Wayland.
- Remote desktop software tends to work better under X.Org Server than Wayland.
- It's easier to recover from crashes under X.Org Server than Wayland.

However, Canonical still expressed its commitment to Wayland in the future.

DESKTOP ENVIRONMENT

A **desktop environment**, also known as a **window manager**, is a client to a display server that tells the server how to draw graphical elements on the screen. In other words, the desktop environment controls the look and feel of the GUI for the entire operating system, providing a common graphical interface for applications running in the environment.

Desktop environments implement the desktop metaphor, in which the user's monitor is treated as if it were the top of a physical desk, where various objects are placed and accessible to the person sitting at the desk. The primary structural element of a desktop metaphor is the application window, and within each window there can exist various graphical elements like buttons, icons, menus, toolbars, and more.



The GNOME desktop environment.

Desktop environments also provide a graphical login screen and can be customized to run every time the system boots. In most cases, the desktop environment will also

come packaged with a variety of different GUI applications, like a file browser, web browser, text editor, and more.

Note: Some desktop environments can run on X, Wayland, or both.



SPECIFIC DESKTOP ENVIRONMENTS

There are many desktop environments available for Linux, the most common of which are described in the following table.

Desktop Environment	Description
GNOME	<p>This is the default desktop environment in most Linux distributions. It can run on both X and Wayland. GNOME follows design principles called human interface guidelines (HIG) that emphasize clarity of focus, simplicity, limiting the amount of work the user has to perform, and many more. GNOME also offers a great deal of support for people with disabilities.</p> <p>The GNOME Shell is the actual user interface used by the desktop environment, and it defines the placement and look of elements like toolbars, menus, and more. Starting with version 3, GNOME changed from using a typical desktop metaphor to a more abstract metaphor in which users can switch between virtual desktops and tasks from an overview screen.</p> <p>There is also the GNOME Classic shell, which provides a user interface similar to GNOME version 2 even with version 3 installed. Canonical also developed the Unity shell for GNOME on Ubuntu, but switched to using GNOME 3 in 2018. Unity was initially designed to make better use of space on devices with small screens, such as netbooks. For example, the app launcher on the left side of the screen is always present in Unity—requiring a single click to launch an app—whereas it requires keyboard input or moving the mouse to the top-left corner of the screen in order to appear in the GNOME Shell.</p>
KDE Plasma	<p>While not as popular as GNOME, KDE® Plasma is the second-most common desktop environment and is included in distributions like RHEL and CentOS®, even if not set as the default. As of 2018, the current version is KDE Plasma 5.</p> <p>KDE Plasma supports modularity through widgets, enabling users to add, move, and remove screen elements to fit their own personal workflow. KDE Plasma also has a robust set of GUI apps, including word processors, mail clients, multimedia software, and more. It supports both X and Wayland.</p>
Cinnamon	<p>This is actually a fork of GNOME 3 and is one of the default environments for the Linux Mint distro. Cinnamon was developed in response to the changes in GNOME 3, and uses a typical desktop metaphor rather than an abstract one. It is therefore similar to GNOME 2. However, as of 2018, Cinnamon does not support Wayland.</p>

Desktop Environment	Description
MATE	This is another fork of GNOME that was created in response to the changes in GNOME 3. It was developed to maintain and extend the functionality of GNOME 2. Many of its GUI applications are forked versions of the GNOME Core Applications. MATE is the other default environment for Linux Mint, and is also available as an option in some other distributions. Like Cinnamon, it does not currently support Wayland.

CHOOSING THE RIGHT DESKTOP ENVIRONMENT

None of these desktop environments is an objectively "right" choice. Which environment you choose will ultimately come down to personal preference and your comfort level with each. You need to try each environment in order to know which is best for you.

REMOTE DESKTOP

Remote desktop is a concept in which a client connects to a remote system over a network, and is able to sign in to and use the desktop environment of that system. Remote desktop sessions are useful in situations where you must configure a server that has a GUI, but you aren't physically located at that system. In addition, remote desktop sessions are used in an environment where the user connects to and works with their remote computer as if it were located in front of them—in other words, the remote session becomes their primary desktop.

The client computer typically constructs the remote session in its own window, and, when that window has focus, any input on the local client (e.g., a keystroke) gets translated and sent to the remote desktop as if the keystroke were being performed on that system directly.

REMOTE DESKTOP SOFTWARE

There are many software packages available for Linux that enable remote desktop sessions. Software typically comes in two forms: a client application that you install on the computer you're physically located at, and a companion server application that you install on the remote system you're trying to connect to.

Remote Desktop Software Description	
Virtual Network Computing (VNC)	VNC is a cross-platform remote desktop service that enables full remote control of a desktop environment. VNC leverages the Remote Frame Buffer (RFB) protocol. A VNC server must be installed on the target machine, which you can access with a corresponding client. There are many different implementations of VNC that work on Linux, including RealVNC, TightVNC, TigerVNC, and more. VNC can work with X and Wayland.
xrdp	xrdp is a free and open source utility that constructs a Remote Desktop Protocol (RDP)-like server for non-Windows systems. RDP was developed by Microsoft® and is the default remote desktop software on Windows®. So, you can install xrdp on a remote Linux server and use a tool like <code>rdesktop</code> or even the default RDP client to connect to that server—in other words, the client can be on any platform. The xrdp utility is designed to work with X.

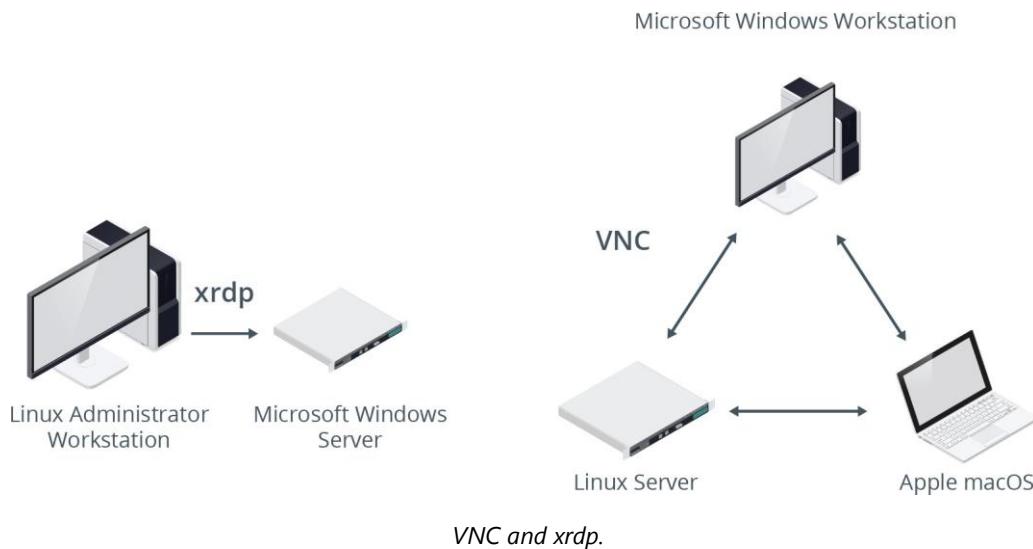
Remote Desktop Software Description

NoMachine (NX)

NX is cross-platform proprietary remote desktop software that offers support for multi-session environments and account management. Therefore, it is useful in organizations with many users that require simultaneous access to the same servers. NX is designed to work with X.

Simple Protocol for Independent Computing Environments (SPICE)

SPICE is a free and open source protocol designed specifically for use in virtual environments. SPICE is often used by administrators to connect to virtual machines that are hosted by the Kernel-Based Virtual Machine (KVM) hypervisor.



VNC and xrdp.

CONSOLE REDIRECTION

Console redirection is the process of forwarding input and output through a serial connection rather than through any I/O peripherals that are directly attached to the system. This enables the system with console redirection to send display output data along a serial cable and eventually to a remote client. Likewise, the remote client can redirect its keyboard input along a serial connection so that it gets sent to the remote server.

Ultimately, console redirection enables administrators to remotely configure systems in a pre-boot environment like BIOS/UEFI. Without an operating system like Linux loaded, typical methods of remote access like SSH and remote desktop will not be available. Using console redirection, administrators can change the BIOS/UEFI settings of multiple machines over a network, even when those machines have no I/O devices directly attached.

Note: Not all systems support console redirection in the BIOS/UEFI.



SSH PORT FORWARDING

Secure Shell (SSH) is a remote access protocol that encrypts transmissions over a network. It is the most commonly used protocol for accessing the command-line

interface of a Linux server. You can use SSH to issue commands to the server as if you were typing into its CLI directly. It can be used as a tunnel to carry other kinds of network communications securely, including remote desktop traffic.

The process of tunneling an application through SSH to secure it in transmission is called **SSH port forwarding**. There are two main types: local and remote forwarding. In local forwarding, the local client listens for connections on a port, and then tunnels any active connection to a remote server using SSH. One use case for local forwarding is to remotely access a system over the Internet using a protocol like VNC. When you connect to the VNC server with your local VNC client, that traffic (usually over a port like 5900) will be forwarded through SSH, securing it.

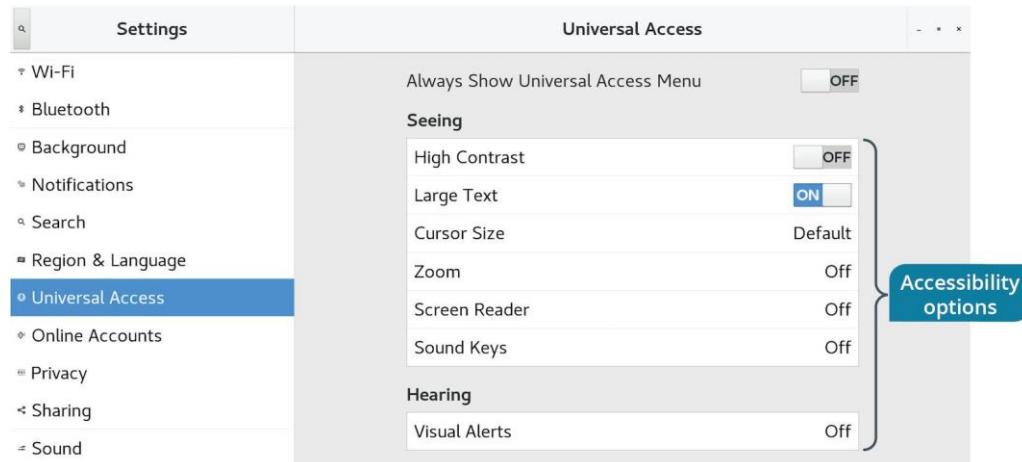
In remote forwarding, the SSH server forwards inbound client traffic to another system on a different port. One use case for remote forwarding is setting up a VNC server on a local system and forwarding all VNC traffic to a port on a remote client. Whenever the remote client connects to this port using localhost (their own network address), they can establish a remote desktop connection with the local server.

X FORWARDING

As you know, X (X11) is network-aware and can enable clients to access GUI elements over a network. You can forward X traffic through an SSH tunnel in order to encrypt these communications.

ACCESSIBILITY OPTIONS

Each desktop environment has its own accessibility options for accommodating people with disabilities.



Configuring accessibility options in GNOME's Universal Access menu.

Note: Accessibility options available in one environment may not exist in another.

Some common options include:

- A screen reader that reads all of the highlighted screen elements.
- A magnifier that zooms in on specific sections of the screen.
- Increasing the size of specific elements like text and the mouse pointer.
- An on-screen keyboard that enables users to click to type text using the mouse, instead of using a physical keyboard.
- Keyboard accessibility options, including:

- Sticky keys, enabling users to press one key at a time instead of holding down multiple keys at once in order to activate a keyboard shortcut.
- Repeat keys, enabling the user to hold down a key in order to repeatedly enter it.
- Toggle keys, enabling the system to beep when keys like **Caps Lock** and **Num Lock** are pressed.
- Visual themes like high contrast that make the screen easier to read for certain people with visual impairments.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 8-3

Discussing GUIs

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **True or false? A graphical desktop environment is required for most administrative activities, such as creating new users and installing new software.**

 2. **What is the primary architectural difference between the X and Wayland display servers? How does this difference affect performance?**

 3. **Where can you, as a user, switch to a different graphical user interface (KDE to GNOME, for example)?**

 4. **At a foundational level, many remote desktop protocols (e.g., NX, xrdp, SPICE, VNC) are not designed to be highly secure. How can you make them more secure?**

 5. **Why would an administrator need or use console redirection?**
-

Activity 8-4

Configuring GUIs

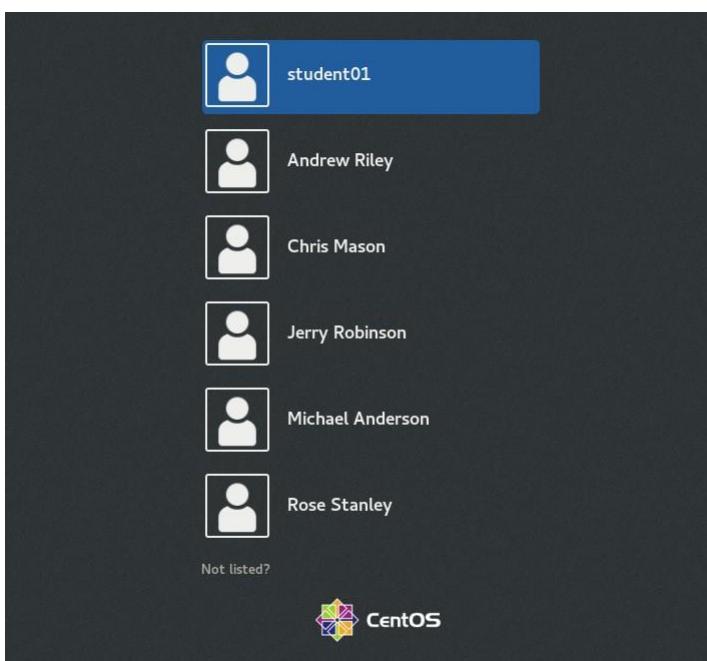
BEFORE YOU BEGIN

You are signed in to the CLI as your student account. The CentOS system was already installed with two GUI desktop environments: GNOME and KDE.

SCENARIO

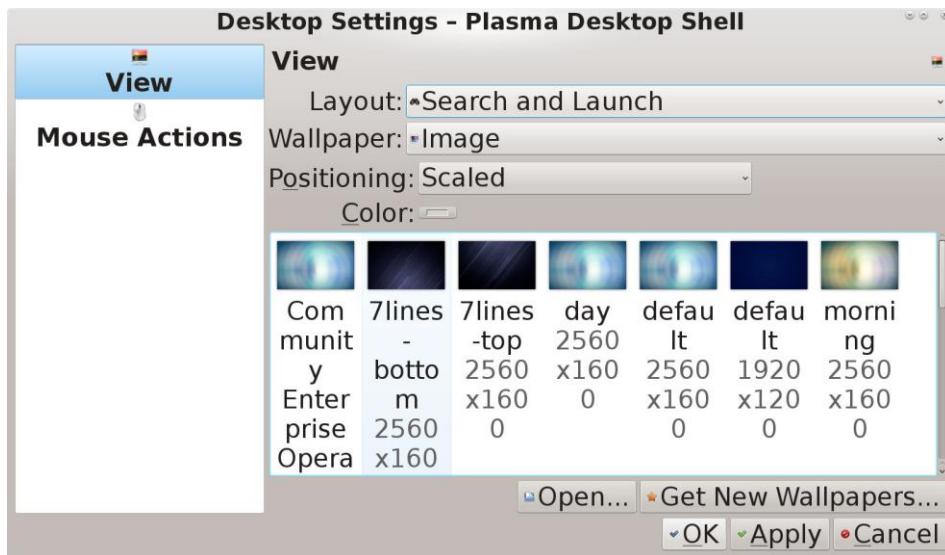
The CLI has been adequate so far, but many of your colleagues would be more comfortable working in a visual environment. A GUI is also necessary for easily browsing the web and viewing media like images and video. Most of your colleagues prefer to work with GNOME, the default desktop environment, whereas others prefer a customized version of KDE. So, you'll start by configuring KDE's layout options to align with those users' preferences. Then, you'll get accustomed to navigating through GNOME, the environment you yourself will be using. You also need to configure some accessibility options in GNOME for users who have visual and manual dexterity impairments.

1. Switch to the GUI, then log in to KDE.
 - a) Enter **Sudo systemctl isolate graphical.target**
 - b) Verify that you are presented with a graphical login screen.



- c) Select **student##**.
- d) To the left of the **Sign In** button, select the **Settings** gear icon.
- e) Select **KDE Plasma Workspace**.

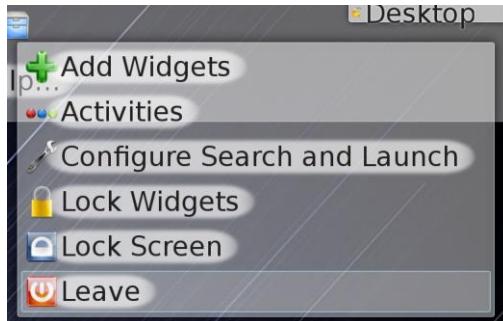
- f) Enter your password and select **Sign In**.
- 2.** Configure desktop settings for KDE.
- On the desktop, right-click and select **Default Desktop Settings**.
 - In the **Desktop Settings - Plasma Desktop Shell** window, in the left pane, verify that **View** is selected.
 - From the **Layout** drop-down list, select **Search and Launch**.
 - In the images section, select **7lines-bottom** from the thumbnails list.



- Select **OK** to apply the settings and close the **Desktop Settings - Plasma Desktop Shell** window.
- Examine the new layout of the desktop environment.

3. Switch to GNOME.

- From the top-right of the desktop, select the **Desktop** menu, then select **Leave**.



- Select **Logout**.
 - Select **student##**.
 - To the left of the **Sign In** button, select the **Settings** icon.
 - Select **GNOME Classic**.
 - Sign in using your password.
- 4.** Go through the preliminary setup options.
- On the **Welcome** screen, select **Next**.
 - On the **Typing** screen, select **Next**.
 - On the **Network** screen, select **Skip**.

- d) On the **Privacy** screen, select **Next**.
 - e) In the dialog box, select **Deny Access**.
 - f) On the **Online Accounts** screen, select **Skip**.
 - g) On the **Ready to Go** screen, select **Start using CentOS Linux**.
 - h) Close the **Getting Started** window.
 - i) Examine the desktop and note that GNOME has a different look and feel than KDE.
5. Navigate the desktop and open an application.
- a) From the desktop menu, select **Applications**→**Favorites**→**Files**.
 - b) Verify that you are in your home directory.
 - c) Double-click the **backup_report** file to open it in the default GUI text editor.
 - d) Make a change to the file, then select **Save**.
 - e) Close the text editor.
 - f) Close the file browser.
6. Enable accessibility settings in GNOME for users with visual and manual dexterity impairments.
- a) Select **Applications**→**System Tools**→**Settings**.
 - b) From the navigation pane on the left, select **Universal Access**.
 - c) In the **Seeing** section, slide the **High Contrast** slider to **On**.
 - d) In the **Typing** section, slide the **Screen Keyboard** slider to **On**.
 - e) Close the **Settings** window.
7. Open a terminal, then revert the accessibility changes.
- a) From the desktop menu, select **Applications**→**Favorites**→**Terminal**.
 - b) Verify that the on-screen keyboard was automatically opened.
 - c) From the top-right of the desktop taskbar, select the **Universal Access** icon , then slide **Screen Keyboard** to **Off**.
 - d) Do the same for **High Contrast**.
8. Keep the terminal window open.
-

Topic C

Manage Services



EXAM OBJECTIVES COVERED

2.4 Given a scenario, manage services.

4.1 Given a scenario, analyze system properties and remediate accordingly.

Services are how the operating system provides functionality to users and applications, and as such, can be managed just as any other part of Linux. In this topic, you'll start and stop services to control available system functionality.

SERVICES AND DAEMONS

A Linux **service** is software that responds to requests from other programs to provide some sort of specialized functionality. Services can be broadly classified as critical services and non-critical services. Critical services are the core services that are vital for the functioning of the Linux system. Non-critical services are services that are initiated by applications installed on the system.

Most services are implemented in the form of **daemons**, which are running programs (i.e., processes) that operate in the background without the need for human intervention. Daemons lie dormant until an event triggers them into activity. Some daemons operate at regular intervals. Most daemons are started when the system boots. Daemons can be started by the operating system, by applications, or manually by the user.



Note: The terms "service" and "daemon" are often used interchangeably. For example, in Windows, a service is the equivalent of a Linux daemon.

SERVICE MANAGEMENT

Service management is the lifecycle process of starting services, modifying their running state, and stopping them. The system can manage services itself, or users can manage services manually depending on their needs. Service management is important in operating systems like Linux because it enables administrators to configure the functionality that their servers provide. It can also help administrators diagnose and troubleshoot issues that affect or are caused by running programs.

SYSTEM INITIALIZATION

System initialization is the process that begins when the kernel first loads. It involves the loading of the operating system and its various components, including the boot process. System initialization is carried out by an **init** daemon in Linux—the "parent of all processes." The init daemon refers to a configuration file and initiates the processes listed in it. This prepares the system to run the required software. The init daemon runs continuously until the system is powered off, and programs on the system will not run without it.

On Linux, there are two main methods that initialize a system: SysVinit and systemd. The method that is active on your Linux system will affect how you manage services on that system.



Note: Here, "init daemon" refers generally to a daemon that performs system initialization, not to the specific SysVInit daemon (named `init`).



Note: Older versions of Ubuntu also had their own init method called Upstart.

THE systemd SUITE

The **systemd** software suite provides an init method for initializing a system. It also provides tools for managing services on the system that derive from the init daemon. The systemd suite was designed as a replacement for other methods like SysVInit, and is now the dominant init method in modern Linux distributions.

The systemd suite offers several improvements over older methods. For example, it supports parallelization (starting programs at the same time for quicker boot) and reduces shell overhead. In systemd, Control Groups (cgroups) are used to track processes instead of process IDs (PIDs), which provides better isolation and categorization for processes.

systemd UNIT FILES

Unit files are configuration files that systemd uses to determine how it will handle units, which are system resources that systemd can manage. Resources can include network services, block storage devices, peripheral devices, file systems, and much more. Daemons access and manage these resources. Resources are defined in one of several categories of unit files, making them easier to manage. A unit file's extension defines its category; e.g., a `.automount` file includes instructions for automatically mounting a mount point defined in a `.mount` unit file. Unit files are written in a declarative language using directives that tell systemd what to do.

```
root@server01 ~]# cat /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.serv
ice
```

Configurations for the graphical.target unit file.

Unit files can exist in multiple locations. The standard location that is used by software to install unit files is the `/lib/systemd/system/` directory. However, you shouldn't edit the unit files in this directory. If you want to modify a unit file's functionality, you should use the `/etc/systemd/system/` directory. Because unit files in this directory take precedence over files elsewhere, you can replace them here. If you want to modify only a portion of the unit file, you can create a directory

named after the unit file with `.d` appended to it, then create a file within this directory that has a `.conf` extension. You can use this `.conf` file to extend or override specific functionality within the unit file.

ENVIRONMENT VARIABLES

Unit files can also be used to set system environment variables/parameters, which are values that are passed from a parent process to any child process it creates. By adding directives for an environment variable, you can make it easier for a service or other unit to work with custom values.

As an example, the `rescue.service` unit file sets the following `Environment` directive:

[Service]

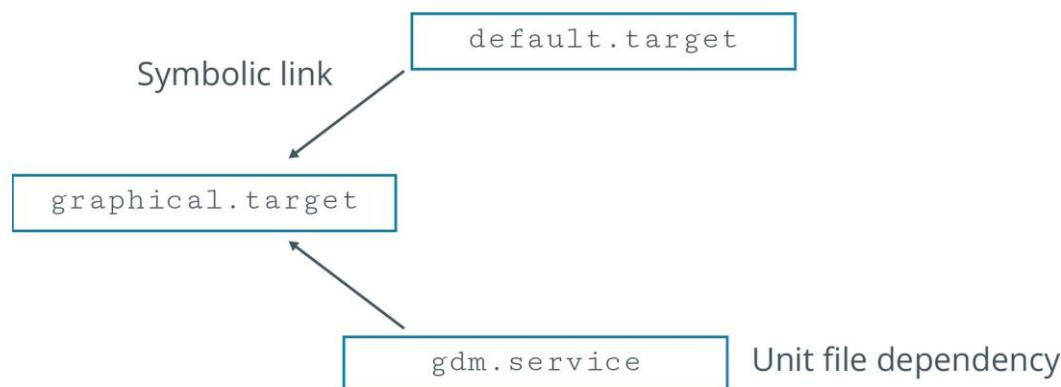
`Environment=HOME=/root`

The `HOME` environment variable is therefore set to the root user's home directory, enabling the rescue mode daemon to log in to the shell with that path as its home directory. When configuring your own unit files, you could use a similar directive, perhaps supplying a different path. You can also set any variable you want—not just `HOME`.

systemd TARGETS

In systemd, **targets** are a method of grouping unit configuration files together, typically to represent specific modes of operation. Each `.target` file is used to determine one of several ways in which the system can operate, such as running with just a CLI; running with a graphical desktop environment; initiating a system shut down; and more. Therefore, you can activate a target in order to boot into the desired environment.

Target files include dependency information that enables these different modes of operation. For example, `graphical.target` will boot the system into a GUI environment. It does this by referencing other unit files like `gdm.service` (the GNOME Display Manager) that are required in order to initialize the GUI and its related components.



An example of how a target file interacts with other systemd components.

THE systemctl COMMAND

The **systemctl** command enables you to control the systemd init daemon. You can view running services, manage (enable/disable) services to run at boot or in the current session, determine the status of these services, and manage the system target.

```
root@server01 ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2019-01-02 18:41:11 G
MT; 5 days ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 1334 (sshd)
    Tasks: 1
   CGroup: /system.slice/sshd.service
           └─1334 /usr/sbin/sshd -D

Jan 02 18:41:11 server01 systemd[1]: Starting OpenSSH serve
```

Checking the status of a service.

You can do all this through various subcommands issued along with the **systemctl** command, as detailed in the following table.

Subcommand	Used To
status {service}	Retrieve the current status of a service.
enable {service}	Enable a service to be started on boot.
disable {service}	Disable a service so that it is no longer started on boot.
start {service}	Start (activate) a service immediately.
stop {service}	Stop (deactivate) a service immediately.
restart {service}	Restart a service immediately.
set-default {target}	Set the default target for the system to use on boot.
isolate {target}	Force the system to immediately change to the provided target.
mask {unit file}	Prevent the provided unit file from being enabled or activated, even when done manually.
daemon-reload	Reload the systemd init daemon, including all unit files.

SYNTAX

The syntax of the **systemctl** command is **systemctl [options] [subcommand] [arguments]**

systemctl COMMAND OPTIONS

The **systemctl** command has many options, some of which are described in the following table.

Option	Used To
<code>-t {unit file type}</code>	Specify the unit file types to perform the operation on.
<code>-a</code>	List all unit files or properties, regardless of state.
<code>--no-reload</code>	Prevent the reloading of configuration changes when enabling or disabling a service.
<code>--no-ask-password</code>	Prevent users from being asked to authenticate when performing privileged operations.
<code>--runtime</code>	Make changes temporary so that they will not be present after a reboot.
<code>-H {remote host}</code>	Execute the operation on the remote host specified by IP address or hostname.
<code>--no-pager</code>	Prevent the output from being piped into a paging utility.

EXAMPLE OF SWITCHING TARGETS

One example of using the `systemctl` command to change targets is when an administrator needs to work in a GUI for a short period of time. It's common to configure a Linux server to boot by default to the CLI (`multi-user.target`). An administrator might switch to the GUI (`graphical.target`) for a particular task, and then switch the server back to `multi-user.target` when the task is complete. The administrator would then leave the server in the more efficient CLI configuration after the task.

The process can be stepped out as follows:

1. The server is started to the `multi-user.target` by default.
2. The administrator logs on to the server at the CLI and enters the following command to start the GUI: `systemctl isolate graphical.target`
3. The administrator completes their administrative tasks.
4. The administrator enters the following command to return the server to the more efficient CLI configuration: `systemctl isolate multi-user.target`

THE `hostnamectl` COMMAND

In most cases, `systemctl` is used to control services, but there are some additional `systemd` commands that you can use. For example, `hostnamectl` enables you to view the system's network hostname and other information about the system's hardware and the Linux kernel it is running. You can also use this command to change the system's hostname.

The syntax of the `hostnamectl` command is `hostnamectl [options] [subcommand] [arguments]`

For example, to set the hostname to `Server01`:

```
hostnamectl set-hostname server01
```

SysVInit AND RUNLEVELS

SysVInit is an older init method that has been largely replaced by `systemd`. However, some distributions still support SysVinit. Like `systemd`, SysVinit provides you with various tools to manage services and the state of the system.

Aside from systemd's improvements, one major difference between it and SysVinit is that SysVinit has **runlevels**. Runlevels control the state of the operating system in much the same way that systemd targets do; they determine what types of daemons should be running in order to create a specific type of environment. In fact, systemd targets were created to map to existing runlevels. Like with systemd targets, you can change a system's runlevel and set a default.



Note: SysVinit was popularized in the UNIX System V family of operating systems, pronounced "System Five."

RUNLEVELS VS. TARGETS

The following table compares SysVinit runlevels with their equivalent systemd targets.

SysVinit Runlevel	systemd Target	Description
0	poweroff.target	Halts (shuts down) the system.
1	rescue.target	Starts single-user mode.
2	multi-user.target	Starts multi-user mode without remote networking. Loads a command-line interface (CLI).
3	multi-user.target	Starts multi-user mode with remote networking. Loads a CLI.
4	multi-user.target	Not used.
5	graphical.target	Starts multi-user mode with networking and GUI capabilities. Loads a desktop environment.
6	reboot.target	Reboots the system.

SINGLE-USER MODE

Single-user mode boots the operating system into an environment where the superuser must log in. Networking is also disabled in single-user mode, and most partitions are not mounted. It is typically used to troubleshoot issues with networking or issues that prevent you from booting into the normal multi-user environment.

On some systems, particularly those using the original GRUB bootloader, it is possible to reset the root password from single-user mode in case it is lost. However, with GRUB 2, you must edit the bootloader configuration at the boot menu so that it initializes a privileged shell that doesn't require a password. From this shell, you can reset the root user's password.

THE telinit AND runlevel COMMANDS

The **telinit** command enables you to switch the current runlevel of the system. On systemd environments, the **telinit** command will be translated into the appropriate target request. The **runlevel** command prints the previous and current runlevel of the system, each separated by a space.

SYNTAX

The syntax of the **telinit** command is **telinit [options] {runlevel}**

THE /etc/inittab FILE

The **/etc/inittab** file stores details of various processes related to system initialization on a SysVinit system. It also stores details of the runlevels in use. The init daemon reads from this file to determine what runlevel to boot into, what daemons to start, and what to do if the runlevel changes. Each entry in the **/etc/inittab** file takes the format:

id:rstate:action:process

The **id** is just a unique identifier for the entry; **rstate** defines what runlevels the entry applies to; and **action** specifies one of several tasks that determine how SysVinit handles the command defined in the **process** field.

THE /etc/init.d/ DIRECTORY

The **/etc/init.d/** directory stores initialization scripts for services. These scripts control the initiation of services in a particular runlevel. The scripts are invoked from the **/etc/inittab** file when the system initialization begins, using the symbolic links found in the file. SysVinit scripts are highly flexible and can be configured according to the needs of the user.

Depending on the distribution, SysVinit scripts may instead be stored in **/etc/rc.d/** or this directory may contain symbolic links to the **/etc/init.d/** directory.

THE /etc/rc.local FILE

The **/etc/rc.local** file is executed at the end of the init boot process, typically used to start custom services. However, this file is rarely used and is not even supported in some distributions that still use SysVinit.

THE chkconfig COMMAND

The **chkconfig** command can be used to control services in each runlevel. It can also be used to start or stop services during system startup. The following are some subcommands and options that can be used with **chkconfig** to control services.

Subcommand/Option	Used To
{service} on	Enable a service to be started on boot.
{service} off	Disable a service so that it is no longer started on boot.
{service} reset	Reset the status of a service.
--level {runlevel}	Specify the runlevel in which to enable or disable a service.

SYNTAX

The syntax of the **chkconfig** command is **chkconfig [options] [service] [subcommand]**

THE service COMMAND

The **service** command is another way to control SysVinit services through SysVinit scripts. It supports the following subcommands.

Subcommand	Used To
{service} status	Print the current state of a service.
{service} start	Start (activate) a service immediately.
{service} stop	Stop (deactivate) a service immediately.
{service} restart	Restart a service immediately.
{service} reload	Re-read a service's configuration files while the service remains running.

SYNTAX

The syntax of the **service** command is **service [options] [service] [subcommand]**



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 8-5

Discussing Services

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **After installing the Apache web server, httpd, you want to check its status. How can you check the status of the httpd service?**

 2. **How can you activate a service such as httpd?**

 3. **You want to be sure that the Apache web server starts on boot. Which command can you use to ensure that it starts?**

 4. **What is the SysVinit equivalent of systemctl enable httpd.service?**

 5. **You made a change to the /etc/httpd/conf/httpd.conf file and saved it, but the change you made hasn't taken effect. What is the problem and how do you fix it?**
-

Activity 8-6

Managing Services

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. A terminal window is open. If at any time your system is locked and shows an empty black screen, press **Spacebar** and then enter your password to log back into the GUI.

SCENARIO

As a Linux administrator at Develetech, you know that you will be implementing, managing, and reconfiguring different services. You'll be leveraging systemd, and in particular, the **systemctl** command and its associated subcommands, to manage these services. You decide to start by switching targets from CLI to GUI, and then making the default target GUI so that users will always boot into that environment by default. Then, you'll practice managing the SSH and firewall services by putting them through the service management lifecycle of starting, stopping, enabling, and disabling them.

1. Verify that your system is using systemd and not the older SysVinit method.
 - a) In the terminal window, enter **ps -e | grep -i init** to check for the **init** process.
 - b) Verify that the **init** process was not found.
 - c) Enter **ps -e | grep -i systemd** to check for the **systemd** process.
 - d) Verify that the **systemd** process was found and has a process ID of 1.

```
[student01@localhost ~]$ ps -e | grep -i systemd
  1 ?          00:00:38 systemd
  511 ?        00:00:06 systemd-journal
  540 ?        00:00:01 systemd-udevd
  820 ?        00:00:02 systemd-logind
```

2. View the target files that specify the services that will start when the system starts.
 - a) Enter **cat /usr/lib/systemd/system/multi-user.target**
Observe that the **multi-user.target** requires the **basic.target**— i.e., the target files build upon each other.
 - b) Enter **cat /usr/lib/systemd/system/graphical.target**
Observe that the **graphical.target** requires the **multi-user.target**— this further illustrates how the target files build on each other. In addition, the **graphical.target** calls or "wants" the **display-manager.service**, which initiates the GUI.
 - c) Enter **Systemctl --type=service** to view the current target's services.

- d) Press **q** when you're finished.
- 3.** Switch between the CLI target and the GUI target, then set the GUI target as the default.
- Enter **sudo systemctl isolate multi-user.target** to switch back to the command-line interface.
 - Sign in as **student##**
 - Enter **sudo systemctl isolate graphical.target** to switch to the graphical user interface.
 - Sign in as **student##** and open a terminal.
 - Enter **sudo systemctl set-default graphical.target** to set the GUI as the default environment.

4. Examine the **systemctl** subcommands.

- a) Type **systemctl** and then type a space, and then press **Tab** twice.

You can use this trick with some commands to list all of their available subcommands.



Caution: Be sure to add a space after the command and before pressing Tab.

- b) Note the **stop**, **start**, **restart**, **status**, **enable**, and **disable** subcommands in particular.

5. Manage the SSH service on your server.

- a) Enter **sudo systemctl status sshd.service**
- b) Verify that the **sshd** service is running.

```
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor
   preset: enabled)
     Active: active (running) since Tue 2018-12-11 18:31:12 GMT; 3 days ago
       Docs: man:sshd(8)
              man:sshd_config(5)
      Main PID: 1346 (sshd)
         Tasks: 1
        CGroup: /system.slice/sshd.service
                  └─1346 /usr/sbin/sshd -D
```

- c) Enter **sudo systemctl stop sshd.service**

- d) Verify that the **sshd** service is stopped.

```
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor
   preset: enabled)
     Active: inactive (dead) since Fri 2018-12-14 19:48:49 GMT; 4s ago
       Docs: man:sshd(8)
              man:sshd_config(5)
      Process: 1346 ExecStart=/usr/sbin/sshd -D $OPTIONS (code=exited, status
 =0/SUCCESS)
     Main PID: 1346 (code=exited, status=0/SUCCESS)
```

- e) Enter **sudo systemctl start sshd.service** to start the **sshd** service again, then check its status to ensure it is running.

6. Disable, and then re-enable, the **firewalld** service.

- a) Enter **sudo systemctl status firewalld.service**
 - b) Verify that the **firewalld** service is running.
 - c) Enter **sudo systemctl disable firewalld.service**
 - d) Verify that the **firewalld** service is still running.
The **enable** and **disable** subcommands do not affect the current status of the service, but rather the startup status.
 - e) Reboot the server, log back in to the GUI with your **student##** account, and verify that the **firewalld** service is not running.
 - f) Start the **firewalld** service again.
 - g) Use the **systemctl** command to enable the **firewalld** service.
-

Topic D

Troubleshoot Process Issues



EXAM OBJECTIVES COVERED

- 2.4 Given a scenario, manage services.
- 2.6 Given a scenario, automate and schedule jobs.
- 4.2 Given a scenario, analyze system processes in order to optimize performance.

During operation of the system, you may encounter various issues that degrade performance or otherwise make the system unusable. In many cases, these issues impact services, daemons, and other instances of running software. So, in this topic, you'll switch from managing running software to troubleshooting problems that are caused by or affect that software.

COMMON PROCESS ISSUES

There are many possible issues that could affect or be caused by a process, a running instance of software. These issues may include:

- A process hangs, causing instability in that process.
- A process hangs, consuming resources that should be allocated to other processes.
- A process hangs, causing general system sluggishness and instability.
- A process terminates before it can perform its intended tasks.
- A process fails to terminate when it is no longer needed.
- A process should be allocated most CPU resources but other processes are instead.
- A process is causing the system to take a long time to boot.
- A process has a file open, preventing you from modifying that file.
- A process has spawned multiple child processes that are hard to keep track of.
- A process is unidentifiable or potentially malicious.

PROCESS STATES

Processes go through a lifecycle from creation until eventual termination. There are five different states that a process can be in during this lifecycle, each one defining how the system or other apps can interact with that process. Knowing a process's state can help you determine what to do with that process or help you diagnose problems concerning that process.

The five states are:

- **Running**—The process is currently executing in user space or kernel space. In this state, the process can perform its assigned tasks.
- **Interruptible sleep**—The process relinquishes access to the CPU and waits to be reactivated by the scheduler. A process typically enters this state when it requests currently unavailable resources. "Interruptible" implies that the process will wake from its sleep if a scheduler finds a time slot for it.
- **Uninterruptible sleep**—In this sleep state, the process will only wake when the resource it's waiting for is made available to it. Otherwise, it will stay in its sleep state. This state is common for processes that perform storage or network I/O.
- **Zombie**—This state indicates that a process was terminated, but that it has not yet been released by its parent process. It is in a "zombie-like" state where it cannot accept a kill signal because the process isn't available anymore.

- **Stopped**—This state indicates that the process was stopped by a debugger or through a kill signal.

PROCESS IDs

Every process is assigned a unique **process ID (PID)** when it is started so that the system and users can identify the process. This PID is a non-negative integer that increases for each new process that is started. The init daemon always has a PID of 1 because it is the first process to start and is the parent of all other processes on the system. Processes started after this, whether by the system or by the user, are assigned a higher available number.

PID	CMD
1	systemd
2	kthreadd
3	ksoftirqd/0
5	kworker/0:0H
7	migration/0
8	rcu_bh
9	rcu_sched

Process IDs.

When it comes to troubleshooting, you'll need to know a process's PID in order to terminate it, change its priority, and perform other management tasks on it.

THE pgrep COMMAND

The **pgrep** command displays the PID of processes that match any given pattern, similar to how **grep** is used to match patterns in a text file. Patterns can include: the name or user ID (UID) of the user who invoked it; the start time; the parent PID; and more.

```
[root@server01 ~]# pgrep sshd
1334
```

Searching for the PID of a specific process.

You can use `pgrep` to help you identify a process based on multiple factors when you don't know its exact PID. Identifying the PID is typically the first step in managing the process.

SYNTAX

The syntax of the `pgrep` command is `pgrep [options] {pattern}`

THE ps COMMAND

The `ps` command invokes the **process table**, a record that summarizes the current running processes on a system. When the command is run without any option, it displays the processes run by the current shell with details such as the PID, the terminal associated with the process, the accumulated CPU time, and the command that started the process. However, different options may be used along with the command to filter the displayed fields or processes.

```
[root@server01 ~]# ps -e
  PID TTY          TIME CMD
    1 ?        00:01:42 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:02 ksoftirqd/0
    5 ?        00:00:00 kworker/0:0H
    7 ?        00:00:02 migration/0
    8 ?        00:00:00 rcu_bh
    9 ?        00:05:55 rcu_sched
   10 ?       00:00:00 1ru-add-drain
```

Listing all processes on the system. Note that a question mark indicates that a process has no controlling terminal.

SYNTAX

The syntax of the `ps` command is `ps [options]`

ps COMMAND OPTIONS

The `ps` command supports several options. Some of the more prominent options are listed here.

Option	Used To
a	List all user-triggered processes.
-e	List all processes.
-l	List processes using a long listing format.
u	List processes along with the user name and start time.
r	Exclude processes that are not running currently.
x	Include processes without a terminal.

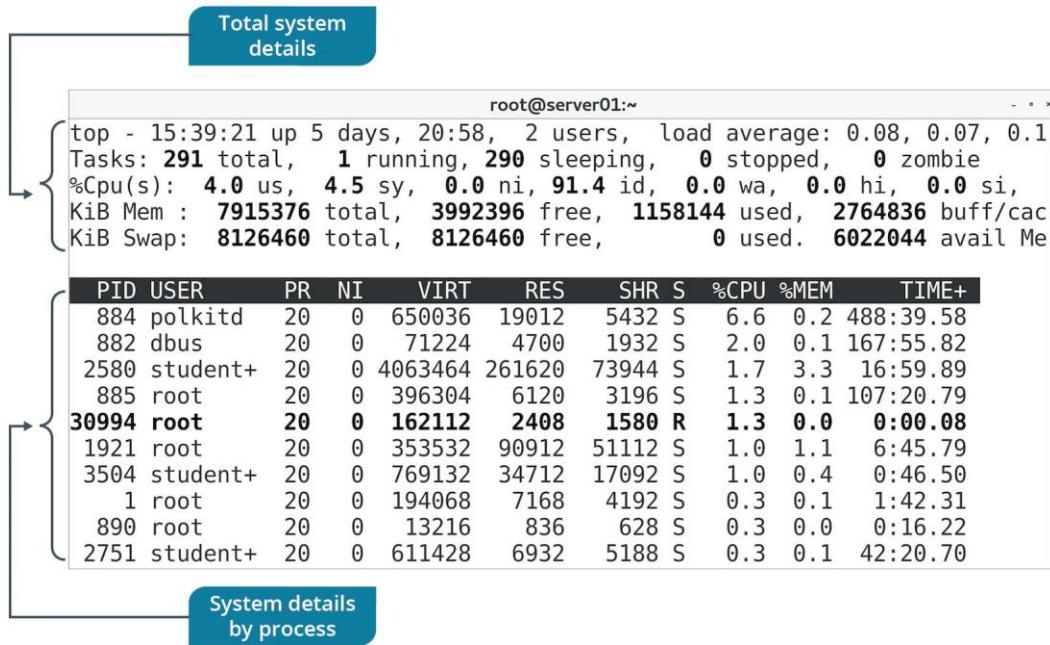
Option	Used To
-T	Exclude processes that were started by any terminal other than the current one.
-U {user name}	Display the processes based on the specified user.
-p {PID}	Display only the process associated with the specified PID.
-C {command}	Display all processes by command name.
--tty {terminal number}	Display all processes running on the specified terminal.

COMMAND OPTION SYNTAX STYLES

The `ps` command supports three different styles of command option syntax: Unix-style (preceded by a hyphen), GNU-style (preceded by two hyphens), and BSD-style (no hyphen). Mixing these styles will not always produce the same results. For example, the `ps a` command (BSD-style) will print all processes with a controlling terminal, including session leaders (the first member of a group of processes). It will also print the status of each process, as well as the full command (including options) of each process. The `ps -a` command (Unix-style) also prints all processes with a controlling terminal, but does not include session leaders, the status of each process, nor the full command of each process.

THE top COMMAND

Like `ps`, the `top` command lists all processes running on a Linux system. It acts as a process management tool by enabling you to prioritize, sort, or terminate processes interactively. It displays a dynamic process status, reflecting real-time changes.



The screenshot shows the `top` command output on a Linux system. At the top, it displays "Total system details". Below that, the system load average is shown as "root@server01:~" followed by "0.08, 0.07, 0.1". The output then provides system statistics: "Tasks: 291 total, 1 running, 290 sleeping, 0 stopped, 0 zombie", "%Cpu(s): 4.0 us, 4.5 sy, 0.0 ni, 91.4 id, 0.0 wa, 0.0 hi, 0.0 si", and memory usage: "KiB Mem : 7915376 total, 3992396 free, 1158144 used, 2764836 buff/cac" and "KiB Swap: 8126460 total, 8126460 free, 0 used. 6022044 avail Me". The main part of the output is a table of processes, with the first few rows highlighted. The table columns are: PID, USER, PR, NI, VIRT, RES, SHR, S, %CPU, %MEM, and TIME+. The highlighted row shows process 30994 with the user root, priority 20, nice value 0, and the command "162112 2408 1580 R 1.3 0.0 0:00.08". Other processes listed include polkitd, dbus, student+, root, and several others with their respective details.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+
884	polkitd	20	0	650036	19012	5432	S	6.6	0.2	488:39.58
882	dbus	20	0	71224	4700	1932	S	2.0	0.1	167:55.82
2580	student+	20	0	4063464	261620	73944	S	1.7	3.3	16:59.89
885	root	20	0	396304	6120	3196	S	1.3	0.1	107:20.79
30994	root	20	0	162112	2408	1580	R	1.3	0.0	0:00.08
1921	root	20	0	353532	90912	51112	S	1.0	1.1	6:45.79
3504	student+	20	0	769132	34712	17092	S	1.0	0.4	0:46.50
1	root	20	0	194068	7168	4192	S	0.3	0.1	1:42.31
890	root	20	0	13216	836	628	S	0.3	0.0	0:16.22
2751	student+	20	0	611428	6932	5188	S	0.3	0.1	42:20.70

Total system details

System details by process

Listing the state of running processes.

Different keystrokes within this tool execute various process management actions. Some of the frequently used command keys include the following.

Key	Used To
E nter	Refresh the status of all processes.
S hift+N	Sort processes in the decreasing order of their PID.
M	Sort processes by memory usage.
P	Sort processes by CPU usage.
u	Display processes belonging to the user specified at the prompt.
k	Terminate the process for which you specify the PID.
r	Renice the process for which you specify the PID.
q	Exit the process list.

SYNTAX

The syntax of the **top** command is **top [options]**

THE **systemd-analyze** COMMAND

The **systemd-analyze** command is used to retrieve performance statistics for boot operations. The command takes one or more subcommands that determine what type of information to print, and how. For process management and troubleshooting, **blame** is the most relevant subcommand. This will print a list of all systemd units that were executed at boot, along with the time it took each unit to execute. You can use **systemd-analyze blame** to identify services and other units that make the system slow to boot.

```
root@server01 ~]# systemd-analyze blame
             → 6.643s systemd-udev-settle.service
                 2.658s kdump.service
                 1.852s dev-mapper-centos\x2droot.device
                 1.768s NetworkManager-wait-online.service
                 1.572s lvm2-pvscan@8:4.service
                 1.491s lvm2-monitor.service
                 1.359s lvm2-pvscan@8:5.service
                 1.194s postfix.service
                 1.175s lvm2-pvscan@8:3.service
                 1.156s tuned.service
                 1.064s vdo.service
                 697ms network.service
                 564ms NetworkManager.service
```

Analyzing how long it took services to execute on boot.

SYNTAX

The syntax of the **systemd-analyze** command is **systemd-analyze [options] [subcommand]**

THE **lsof** COMMAND

The **lsof** command prints a list of all files that are currently opened to all active processes. This can include everything from a text file to a device file—any object that

the system can parse as a file. You may be prevented from modifying a file if it is opened in another process. By using **lsof** you can identify the offending process for termination. You can also use **lsof** to analyze how a process uses files, which can be helpful in identifying malicious processes or processes that have unwanted side effects.

The **lsof** command prints each file opened by a process on its own line. It prints information such as:

- The name of the command/process.
- The PID.
- The invoking user.
- The file descriptor (FD), including what permissions the file is open with.
- The type of file.
- The name of the file.

```
root@server01 ~]# lsof +d ~
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
      Output information may be incomplete.
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
lsof    10481 root cwd   DIR 253,0    4096 100663393 /root
lsof    10482 root cwd   DIR 253,0    4096 100663393 /root
bash   10529 root cwd   DIR 253,0    4096 100663393 /root
```

Listing all processes that have the home directory open.

SYNTA

X

The syntax of the **lsof** command is **lsof [options]**

PROCESS PRIORITIES

The OS provides a scheduler that determines what processes to give CPU time to. The scheduler is usually effective at assigning CPU time, but not every decision it makes is optimal. In some cases, you may need to manually tell the CPU to prioritize certain processes over others. It may be more crucial for one process to get done quickly over another, less-crucial process; you may want to de-prioritize processes started by unprivileged users; a process may be consuming too many resources and causing system sluggishness; etc.

Processes are prioritized based on a number from -20 to 19, called a **nice value** or niceness value. The lower the number, the higher the priority. A process with a nice value of -18 will be more likely to be given CPU time than a process with a nice value of 15. A process inherits a nice value from its parent, and by default, that value is 0.

THE **nice** COMMAND

The **nice** command enables you to run a command with a different nice value than the default. The **-n** option increments the nice value by the given integer; if you don't

provide an integer, then the command will assume an increment of 10. By running `nice` without any options, you'll see the default nice value. You must have the root user authority to run a command at a higher priority. Once lowered, the priority for any process cannot be increased by normal users, even if they own the process.

SYNTAX

The syntax of the `nice` command is `nice [-n {nice value increment}] [command]`

THE `renice` COMMAND

Whereas `nice` is used to start a new process, the `renice` command enables you to alter the scheduling priority of an already running process. You use the `-n` option to specify the new nice value that you want the process to have.

When you renice a process group with the `-g` option, it causes all processes in the process group to have their nice value altered. When you renice a user with the `-U` option, it alters the nice value of all processes owned by the user. By default, the processes affected are specified by their PIDs.

SYNTAX

The syntax of the `renice` command is `renice [-n {nice value}] [options] {identifier}`

FOREGROUND AND BACKGROUND PROCESSES

In Bash, most commands will be executed in the foreground. The effect of this is that the command prompt will be "consumed" until the command finishes processing. For simple commands, like the `date` command, this isn't a problem. For longer commands or for scripts, this can be an issue. For example, if you execute a backup script that you know will take 45 minutes to complete, you will not be able to enter a command in that shell until the script has finished.

It is possible to run commands in the background, where they still execute but do not consume the shell. You can use the `fg` (foreground) and the `bg` (background) commands to move the process into view.

MANAGEMENT OF FOREGROUND AND BACKGROUND PROCESSES

The following is an example of how to manage a long-running script:

1. First, you would execute the script in the background by entering `backup-script.sh &`
2. Next, you would use the `jobs` command to discover the job ID number of the script. This command would also show you the current status of the job (running, for example).
3. You could move the script's execution into the foreground by entering the `fg %<job ID>` command. This might be useful if the script includes interactive prompts.
4. You then press **Ctrl+Z** to temporarily pause the job, freeing the command prompt.
5. You move the script back into the background by entering `bg %<job ID>`

COMMAND SUMMARY

The following table summarizes the purpose of each command used in the previous example.

Command	Used To
<code>fg %{job ID}</code>	Bring a job to the foreground.
<code>Ctrl+Z</code>	Halt a job temporarily so you can use the <code>bg</code> command.
<code>bg %{job ID}</code>	Push a job to the background.
<code>&</code>	Start a command running in the background when added to the end of a command.

THE nohup COMMAND

The `nohup` ("no hangup") command prevents a process from ending when the user logs off. For example, if an administrator launches a backup script, and then logs off the system, the script would stop running. By placing the `nohup` command in front of the normal command, the script would continue even after the administrator logged off.

SYNTAX

The syntax of the `nohup` command is `nohup {command/script}`

KILL COMMANDS

Different commands are used to send signals to processes to terminate or "kill" them. This is necessary when a process becomes unresponsive (hangs), causes system instability, or fails to relinquish control over a file you're trying to modify.

Command	Description
<code>kill</code>	Sends any specified signal, or by default the termination signal, to one or more processes. The PID must be specified as the argument. The syntax of this command is <code>kill [options] {PID}</code>
<code>pkill</code>	Sends any specified signal, or by default the termination signal, to processes based on a matching pattern. Similar to the <code>pgrep</code> command, but actually sends a signal instead of printing to stdout. For example, if you start <code>top</code> in one terminal, and then issue <code>pkill top</code> in another terminal, you'll see that <code>top</code> terminates. The command matched a name pattern rather than a process ID. The syntax of this command is <code>pkill [options] {pattern}</code>
<code>killall</code>	Sends any specified signal, or the default termination signal, to all processes matching the name specified. Similar to <code>pkill</code> but has a few functional differences, such as matching process names exactly. The syntax of this command is <code>killall [options] {process name}</code>

USING THE PID NUMBER TO TERMINATE PROCESSES

You can use the **kill** command with the process table to end processes. By entering **kill** followed by the PID, you can terminate specific processes. However, the process table may display processes that do not belong to you. As a user, you can use the **kill** command only with processes that you own. As root, you can kill any processes.

KILL SIGNALS

There are many ways to kill a process, each one mapped to a signal. This signal determines how to kill the process. Some signals are more appropriate than others in certain situations. For example, you may wish to terminate a process gracefully, giving it time to clean up any last activities. However, some processes will ignore this signal or simply fail to terminate in a reasonable amount of time. For that, you may have to issue a more aggressive signal.

There are many different kill signals. Each signal has a name and one or more corresponding number values; you can use either with the **kill** command. Some of the most common signals are described in the following table.

Signa	Value(s)	Used To
SIGINT	2	Interrupt a process from the terminal, enabling it to end gracefully. The signal can be caught or ignored. This is the same as pressing Ctrl+C at a terminal; a process might change this shortcut behavior, however.
SIGKILL	9	Kill the process immediately. The signal cannot be caught or ignored. This is typically used as a last resort.
SIGTERM	15	Terminate a process, enabling it to end gracefully. The signal can be caught or ignored. This is typically sent from kill or another program and not the terminal.
SIGSTOP	17, 19, 23	Pause a process. The signal cannot be caught or ignored. This is typically sent from kill or another program and not the terminal.
SIGSTP	18, 20, 24	Pause a process from the terminal. The signal can be caught or ignored. This is the same as pressing Ctrl+Z at a terminal; a process might change this shortcut behavior, however.

EXAMPLES

The following are some examples of implementing kill signals. To terminate a process with ID 921 gracefully:

kill 15 921

Failing that, to kill the process immediately:

kill 9 921

Alternatively, to pause rather than remove the process entirely:

kill 17 921

GUIDELINES FOR TROUBLESHOOTING PROCESS ISSUES

Use the following guidelines when troubleshooting process issues.

TROUBLESHOOT PROCESS ISSUES

When troubleshooting process issues:

- Gather information about a process, including its process ID (PID) and state.
- Use `ps` to print information on all running processes, including CPU usage. For example, a process consuming over 50% of the CPU may cause performance issues, and is worth investigating further.
- Use `top` to retrieve a dynamic and interactive list of all running processes.
- Use `Systemd-analyze blame` to determine what startup processes are slowing down boot operations.
- Use `lsof` to identify which processes have open files.
- Use `nice` and `renice` to prioritize certain processes over others.
- Use `fg` and `bg` to manage foreground and background processes.
- Use `nohup` to keep a command running even after logging off.
- Use `kill` and its associated commands to terminate problem processes.

Activity 8-7

Discussing Process Issues

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **One of the other system administrators on your team states that a process is now a zombie. What is the characteristic of a zombie process that makes it a problem for administrators?**

 2. **The Linux system that controls your phone system has experienced some problems with memory lately, and you're curious as to what the problem is. Upon further investigation, you note that the system is fully updated and patched. You decide to check running processes to determine if something has gone awry with one of the services. Which command can you use to check running processes?**

 3. **You've had several complaints about the performance of one of your systems. You log in and find that it responds slowly to commands. You want to look at processes that might be consuming too many resources. Which command can you use?**

 4. **While investigating complaints of system "slowness," you use the top command, but aren't seeing any issues from a CPU usage standpoint. You want to check memory usage while still using the top command. Which command key can you use to display processes by memory usage?**

 5. **It's 6:00 P.M. and you need to leave for the day. However, you also need to run an audit script that creates a report of all files that haven't been accessed in more than 180 days. If you log off, the script will stop running. How can you accomplish both—running the script and leaving work?**
-

Activity 8-8

Troubleshooting Process Issues

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

Some users have complained that processes on the Linux server are taking longer than normal to complete. You discover several processes that are not needed are still running and were never successfully terminated. You need to manage the system processes and the processes issued by other users. In addition, you'll see if there are any problem processes that are consuming too many resources or are causing delays in the boot process.

1. View the current running processes and all other running processes.
 - a) Enter **ps** to list only the processes running on the current terminal.
 - b) Verify that only the processes started by your account are listed.
 - c) Enter **ps -e** to list all the processes running on the system.
 - d) Verify that more processes are listed as compared to the output of the standard **ps** command.
2. Discover the process ID number of a process for which you know the name.
 - a) Enter **pgrep sshd**
 - b) Note the process ID number of the **sshd** service.
3. Issue a background command.
 - a) Enter **sleep 300 &** to pause the system for 300 seconds.
 - b) Make note of the PID of the **sleep 300 &** command:

 - c) Enter **ps**
 - d) Verify that the **sleep** command is in the list of running processes.
4. Terminate the command.
 - a) Enter **kill <PID>** where **<PID>** is the **sleep** process ID that you noted earlier.
 - b) Enter **ps**
 - c) Verify that the **sleep** command is not in the list of running processes.
5. Explore more process information with the **top** command.
 - a) Enter **top** to open the process management tool.
 - b) Press **M** to reorganize the output by memory usage.
 - c) Press **P** to reorganize the output by CPU usage.

- d) Press **q** to exit the **top** program.
6. Discover what files are open, and which processes opened them.
- a) Enter **lsof**
 - b) Enter **lsof -u student##** to see files opened by a specific user.
7. View boot performance information.
- a) Enter **systemd-analyze**
The results of the **systemd-analyze** command break the startup process into three parts: how long it took the kernel to start, how long it took the initrd image to load, and how long user startup applications and services took to start. The command also shows the total amount of time the startup took. This information can be used in troubleshooting long startup times.
 - b) Enter **systemd-analyze blame** to see which processes take the longest to start during boot.
 - c) Press **q**.
-

Activity 8-9

Prioritizing Processes

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

You want to back up the local copy of the `/etc/` configuration directory. You expect the copying process to be time-consuming and to continue after you log out of your system. You decide to increase the priority of the process to ensure that it is completed on time.

1. View the nice values of running processes.
 - a) Enter `ps xl | less` to view all processes run by users.
 - b) Examine the processes that have the highest nice value.
 - c) Press **Page Down** until you reach the end of the entire list.
 - d) Press **q** to exit the list.

2. Issue the command to copy files as a background process.
 - a) Enter the following to begin the copy process:


```
for i in {1..100}; do sudo cp -R /etc /backup/sys; done &
```

This issues the copy command 100 times to simulate a long-running task.
 - b) Verify that the PID and the job number are displayed.
 - c) Make note of the PID:

3. Renice the copy process by using the `top` command.
 - a) Enter `Sudo top` to open the process management tool.
 - b) Press **r**.
 - c) Enter the process ID of the copy operation you noted previously.
 - d) Enter **-15** to specify the nice value.
 - e) Press **q** to exit the process list.
 - f) Enter `ls /backup/sys`
 - g) Verify that the files from the `/etc` directory are listed, indicating that the copy process is successful.

Topic E

Troubleshoot CPU and Memory Issues



EXAM OBJECTIVES COVERED

4.1 Given a scenario, analyze system properties and remediate accordingly.

The system needs to access certain hardware resources—particularly, the processor and memory—in order to provide functionality. Problems that affect these resources will cause major disruption to the system. So, in this topic, you'll use various Linux commands to identify and solve issues related to the CPU and RAM.

COMMON CPU ISSUES

There are many possible issues that could affect or be caused by the CPU. These issues may include:

- The CPU is under-performing for the desired level of speed and responsiveness.
- The CPU is being overloaded by too many requests and can't respond in time.
- One or more CPU cores are non-functional and/or exhibiting reduced performance.
- Processes are unable to access enough CPU time to run effectively.
- Processes are consuming too much CPU time, leaving other processes without resources.
- The CPU is spending too much time idle instead of processing.
- The CPU doesn't support features like virtualization or hyperthreading.

THE /proc/cpuinfo FILE

The `/proc/cpuinfo` file contains information about the system's processor. You can use this information to identify characteristics about your CPU that might indicate issues related to performance or lack of support for features.

```
root@server01 ~]# cat /proc/cpuinfo
processor      : 0
vendor_id     : GenuineIntel
cpu family    : 6
model         : 69
model name   : Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz
stepping       : 1
microcode     : 0x20
cpu MHz       : 1279.370
cache size    : 4096 KB
physical id   : 0
siblings       : 4
core id       : 0
cpu cores     : 2
```

Viewing CPU information.

Each logical processor core has its own entry. A CPU might support multithreading, which performs multiple operations simultaneously on a single physical core. Logical cores represent each possible thread. So, a CPU marketed as being a quad-core processor might have eight logical cores.

Some useful fields in the `/proc/cpuinfo` file include:

- `processor`—The number of the logical core, starting with 0.
- `vendor_id` —The CPU manufacturer.
- `model name` —The specific model of CPU.
- `cpu MHz`—The logical core's clock speed, measured out to the thousandths decimal place.
- `cache size` —The CPU's cache size.
- `flags` —Characteristics about the CPU as well as supported features.

CPU-BASED KERNEL PARAMETERS

As you've seen, the `sysctl` command enables you to view kernel parameters at runtime.

THE `uptime` COMMAND

The `uptime` command displays the time from when a system started running. The output of the `uptime` command gives information about the current time, how long the system is running, and how many users are currently logged in.

Most relevant to CPU troubleshooting, however, is the load average field. A CPU's load is expressed as the number of processes that are either using or waiting to use the CPU. It can also include the number of processes in the queue for storage I/O. Using `uptime` you can find the average load over three different periods of time, from left to right in the output: the last 1 minute, the last 5 minutes, and the last 15 minutes.

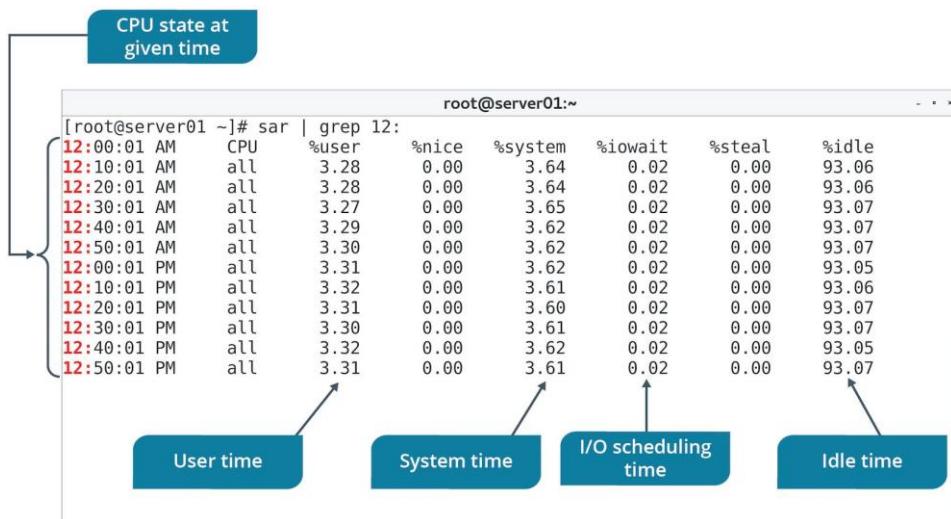


Viewing system uptime and average CPU load.

You can use these values to determine when your CPU becomes overloaded, which may lead to performance issues. For example, if you have 4 logical cores, and the load average in the last 15 minutes is 4.0, then your CPU was being used fully. If your load average were 8.0, then your CPU would be 100% overloaded.

THE sar COMMAND

The **sar** command displays system usage reports based on data collected from system activity. These reports consist of various sections, each of which consists of the type of data and the time at which the data was collected. The default mode of the **sar** command displays CPU usage in various time increments for each category of resource that accessed the CPU, such as users, the system, I/O scheduling, etc. It also displays the percentage of the CPU that was idle at a given time. At the bottom of the report is an average of each data point across the listed time periods. By default, **sar** reports the data collected every 10 minutes, though you can use various options to filter and shape these reports.



Viewing CPU time usage.

Like the **uptime** command, you can use **sar** to identify excessive load on the CPU. You're given more details about when excessive usage occurs, as well as what might be causing that excessive usage.

SYNTAX

The syntax of the **sar** command is **sar [options]**

CPU-BASED KERNEL PARAMETERS

You can also use the **sysctl** command to troubleshoot CPU issues by retrieving CPU-based kernel parameters at runtime. One useful set of parameters concerns scheduling domains, a method by which the kernel groups logical cores that share scheduling policies and other properties. These parameters typically take the format:
kernel.sched_domain.cpu#.domain#.param

COMMON MEMORY ISSUES

There are many possible issues that could affect or be caused by memory. These issues may include:

- Not enough total memory to service all processes at once.
- Not enough free memory to service new processes.
- Processes are unable to access memory despite being available.
- Processes are accessing too much memory, leaving other processes without memory.
- The system cannot quickly access files from a cache or buffer.

- The system is killing critical processes when it is low on memory.
- Swap partitions are taking up too much or not enough storage space.

THE /proc/meminfo FILE

The `/proc/meminfo` file contains a great deal of information about the system's memory usage. You can use this information to ensure that the system's RAM modules are performing to specification; that the OS is consuming memory at the expected rate; and that the system has enough available memory to perform intensive tasks.

Some useful fields in the `/proc/meminfo` file include:

- `MemTotal` —The total amount of physical memory on the system.
- `MemFree` —The total amount of physical memory that is currently unused.
- `Cached` —The total amount of physical memory that is being used as cache memory.
- `SwapTotal` —The total amount of swap space on the system.
- `SwapFree` —The total amount of swap space that is currently unused.
- `Dirty` —The total amount of memory that is waiting to be written to storage.
- `Writeback` —The total amount of memory currently being written to storage.

Note: Most of these values are expressed in kilobytes (KBs).



```
root@server01 ~]# cat /proc/meminfo
MemTotal:      7915376 kB
MemFree:       3989720 kB
MemAvailable:  6022184 kB
Buffers:        12876 kB
Cached:        2472828 kB
SwapCached:     0 kB
Active:        2205036 kB
Inactive:      1215360 kB
Active(anon):   936132 kB
Inactive(anon): 337836 kB
Active(file):  1268904 kB
```

Memory details

Viewing memory information.

THE free COMMAND

The `free` command parses the `/proc/meminfo` file for easier analysis of memory usage statistics. Its default behavior is to display the following information about system memory and swap space:

- The total memory.
- The total used.
- The total free.
- The total shared.
- The total buffered and cached.
- The total available for starting new apps (estimated).

```
root@server01:~#
[root@server01 ~]# free -h
              total        used      free    shared   buff/cache  available
Mem:       7.5G       1.1G     3.8G    341M      2.6G      5.7G
Swap:      7.7G        0B     7.7G
```

**Memory and
swap space info**

Parsing memory and swap space usage information.

SYNTAX

The syntax of the **free** command is **free [options]**

free OPTIONS

There are several command options available for the **free** command.

Option	Used To
-b, -k, -m, -g, -tera	Display memory in bytes, kilobytes, megabytes, gigabytes, and terabytes, respectively.
-s {seconds}	Update memory statistics at a delay of the specified seconds.
-o	Disable the display of the buffered/cached information.
-t	Display a total line that combines physical RAM with swap space.
-h	Make the output more human-readable.

BUFFER/CACHE OUTPUT

Memory can be cached, meaning that it is stored temporarily so that the data it contains can be accessed much quicker in the future. The **Buffers** field in **/proc/meminfo** indicates memory that is assigned to a specific block device. This memory is used to cache file system metadata, like directory contents, permissions, etc. The **Cached** memory is similar, but instead of storing file metadata, it stores the actual contents of files. The **free** command combines these two values together upon output.

Note: When the buffer/cache value is low, the system is low on memory.



THE vmstat COMMAND

The **vmstat** command displays various statistics about virtual memory, as well as process, CPU, and I/O statistics. By default, the report will provide averages of each statistic since the last system boot, though you can also specify a delay value to sample from a period of time.

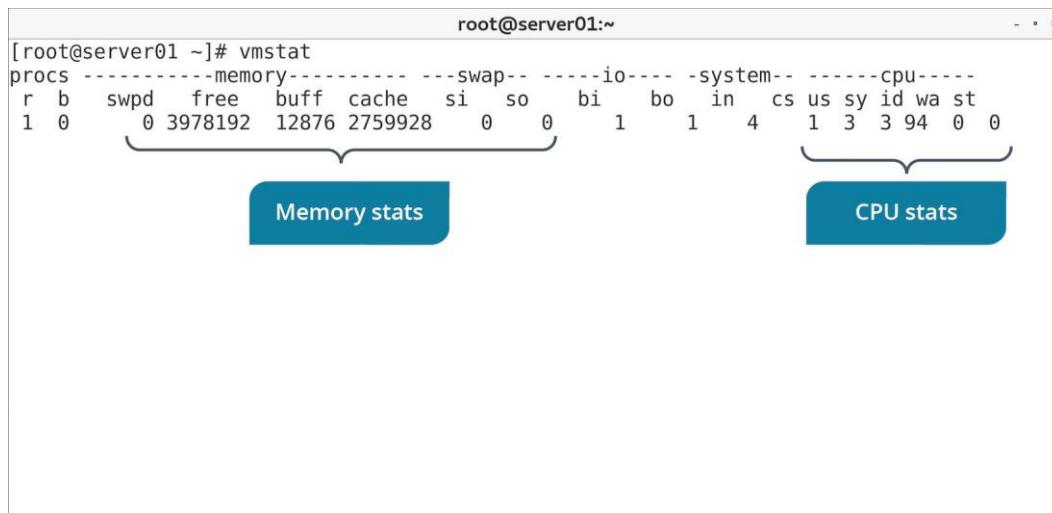
Some memory-based statistics include:

- The total virtual memory available.
- The total virtual memory that is free for use.
- The total memory used in buffers and cache.
- The total memory used in swap space.

Some CPU-based statistics include:

- Time spent running user space.
- Time spent running in kernel space.
- Time spent idle.
- Time spent waiting for I/O.

Note: CPU times are expressed as percentages of total CPU time.



Viewing various memory, CPU, and other system statistics.

SYNTA

X

The syntax of the **vmstat** command is **vmstat [options] [delay [count]]**

OUTPUT DELAY

It's recommended to supply **vmstat** with a delay for a more accurate report. For example, **vmstat 5 5** will run the command on a five-second delay for five intervals.

OOM KILLER

The **out-of-memory (OOM) killer** is a feature of the Linux kernel that determines what process(es) to kill when the system is extremely low on memory. The OOM killer will continue to kill processes until enough memory is free for the kernel and the system to run smoothly. Rather than killing processes at random, the OOM killer leverages an algorithm that assigns each process an OOM score. The higher the score, the higher chance the process has of being killed during an OOM event. The assignment algorithm considers what processes will free up the greatest memory when killed, as well as what processes are least important for system stability. It then assigns scores based on what it determines to be the most optimal targets for termination.

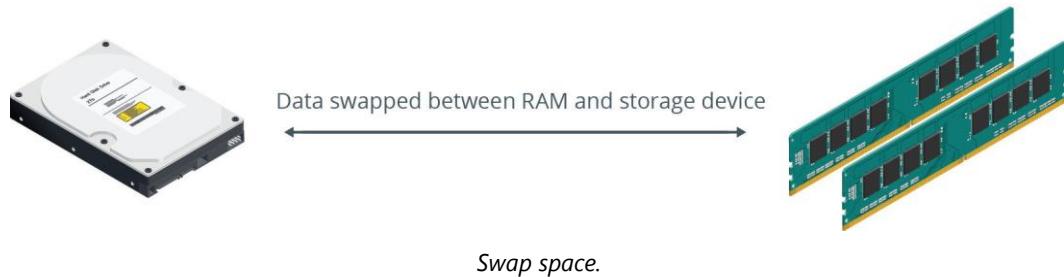
Although this mechanism is mostly automated by the kernel, you do have the ability to control some of its behavior. You can mount the **oom** control group at the desired mount point. If, for example, you want to designate a group of processes as the first to be killed, you can create a directory at this mount point and create a **tasks** file in this directory that lists the PIDs of the relevant processes. Then, create an **oom.priority** file with a high integer value like **256** to give the processes a higher priority for the OOM killer. Or, to prevent the processes from being killed entirely, give **0** as the priority value.



Note: Adding more memory or reducing workload is a more thorough and permanent solution than relying on the OOM killer.

SWAP SPACE CONFIGURATION

The configuration of swap space can alleviate memory-related issues, especially when the system and applications request more memory than the system has. Systems with a low amount of RAM are particularly vulnerable to these issues. Swap space is not a replacement for adding more memory, but it can help minimize system and application sluggishness, unresponsiveness, and crashes.



SWAP SPACE TYPES

Swap space can be one of three types.

Swap Type	Description
Device swap	Device swap space is configured when you partition the storage device. It is used by the operating system to run large applications.
File system swap	File system swap space is configured primarily when you install Linux. It is used by the operating system as an emergency resource when the available swap space runs out.
Pseudo-swap	Pseudo-swap space enables large applications to run on computers with limited RAM.

SWAP FILES

Swap files are created for storing data that is to be transferred from a system's memory to a storage device. They are dynamic and change in size when data is moved in and out of memory. Swap files are used as a medium to transfer data from RAM onto the storage device.

SWAP PARTITIONS

A swap partition is an area of virtual memory on a storage device to complement the physical RAM in the computer. Swap partitions are used by Linux because they perform better than swap file systems.

THE mkswap COMMAND

The **mkswap** command is used to create swap space on a storage partition. It is typically used when you wish to move swap space to a different partition than the one that was created during system installation. For example, you might want to save space on a low-capacity boot drive.

It provides options to perform various tasks.

Option	Used To
-C	Verify that the device is free from bad sectors before mounting the swap space.
-p	Set the page size to be used by the mkswap command. A page is a chunk of memory that is copied to the storage device during the swap process.
-L {label}	Activate the swap space using labels applied to partitions or file systems.

SWAP PARTITION MANAGEMENT COMMANDS

The **swapon** command is used to activate a swap partition in a specified device. The **swapoff** command is used to deactivate the swap space on a device.

Some of the frequently used **swapon** and **swapoff** command options are given in the following table.

Option	Used To
swapon -e	Skip devices that do not exist.
swapon -a	Activate all of the swap space.
swapoff -a	Deactivate all of the swap space.

GUIDELINES FOR TROUBLESHOOTING CPU AND MEMORY ISSUES

Use the following guidelines when troubleshooting CPU and memory issues.

TROUBLESHOOT CPU AND MEMORY ISSUES

When troubleshooting CPU and memory issues:

- Identify key information about the CPU and its logical cores using the **/proc/cpuinfo** file.
- Use the **uptime** command to identify CPU load averages.
- Use **Sar** to see what component is causing heavy load on the CPU and when.
- Identify key information about memory usage using the **/proc/meminfo** file.
- Use **free** to more easily analyze memory usage information.
- Use **vmstat** to retrieve more information on both CPU and memory usage.
- Consider tweaking the OOM killer to spare or sacrifice specific processes when low on memory.
- Consider creating more swap space if adding physical memory is not feasible.

Activity 8-10

Discussing CPU and Memory Issues

SCENARIO

Answer the following questions to check your understanding of the topic.

1. For a quick glance at CPU performance, what does the `uptime` command tell you about the system?
 2. The Performance and Capacity Team in your company requires a system activity report on one of your Linux systems. The team contact has requested a CPU report. Which command can you issue to gather the information for the report?
 3. Some of your users have complained about one of the development servers being slow. With what command can you check the memory statistics that give you the best quick snapshot of memory performance?
 4. The `vmstat` command is useful for displaying an overall performance snapshot, but running it once only displays information since the last reboot. How can you see a more comprehensive view of system performance using the `vmstat` command?
 5. You may configure swap space on a Linux system but not necessarily make it active until you feel like the system requires it. How can you enable a system's configured swap space?

Activity 8-11

Troubleshooting CPU and Memory Issues

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

You want to identify some basic tools to help manage system components. Specifically, you'll review processor and memory usage to see if the results match expected performance.

1. Gather information on the CPU.
 - a) Enter **less /proc/cpuinfo** to display processor information.
 - b) Press **q** to quit.
 - c) Enter **uptime** to see how long the system has been up and view basic performance information on the CPU.
2. Use **sar** to gather system information.
 - a) Enter **Sar -U** to retrieve basic performance information.
 - b) Enter **sar 2 6** to retrieve information every two seconds, for a total of six queries.
 - c) Enter **sar -S** to retrieve swap space usage information.
3. Gather information on system memory.
 - a) Enter **less /proc/meminfo** to display memory information.
 - b) Press **q** to quit.
 - c) Enter **free** to gather basic memory usage information.
 - d) Enter **free -m** to see the memory usage measured in megabytes.
 - e) Enter **free -h** to see memory usage in human-readable format.
4. Use the **vmstat** command to gather virtual memory usage information.
 - a) Enter **vmstat 5 3** and wait for the report to finish.
 - b) Enter **vmstat -d 5 3** to see the information organized on a per-storage device basis.

Summary

In this lesson, you configured system components and managed services. You also engaged in troubleshooting system components like processes, as well as fundamental computing components like the CPU and memory. By managing these crucial elements of a Linux system, you'll be able to ensure optimal performance while also supporting the user experience.

Do you prefer administering your Linux systems in a GUI environment, or just from the CLI? Why? If you prefer working in a GUI, which desktop environment do you prefer to use?

Which process management tools do you expect will be useful in your environment? Why?



Practice Question: Additional practice questions are available on the course website.

Lesson 9

Managing Devices

LESSON TIME: 1 HOUR, 30 MINUTES

LESSON INTRODUCTION

The next component category you can configure is devices. At some point, physical hardware is required to interface with the Linux® operating system, even if you're not co-located with the system itself. You must therefore ensure that any hardware devices connected to the system are recognizable to the system and properly configured. So, in this lesson, you'll manage several different kinds of devices.

LESSON OBJECTIVES

In this lesson, you will:

- Identify the different types of devices that support the Linux OS.
- Configure devices.
- Monitor devices.
- Troubleshoot various issues having to do with hardware devices.

Topic A

Identify the Types of Linux Devices



EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

Before you begin using Linux® commands to manage devices, you need to become familiar with the types of devices that Linux supports.

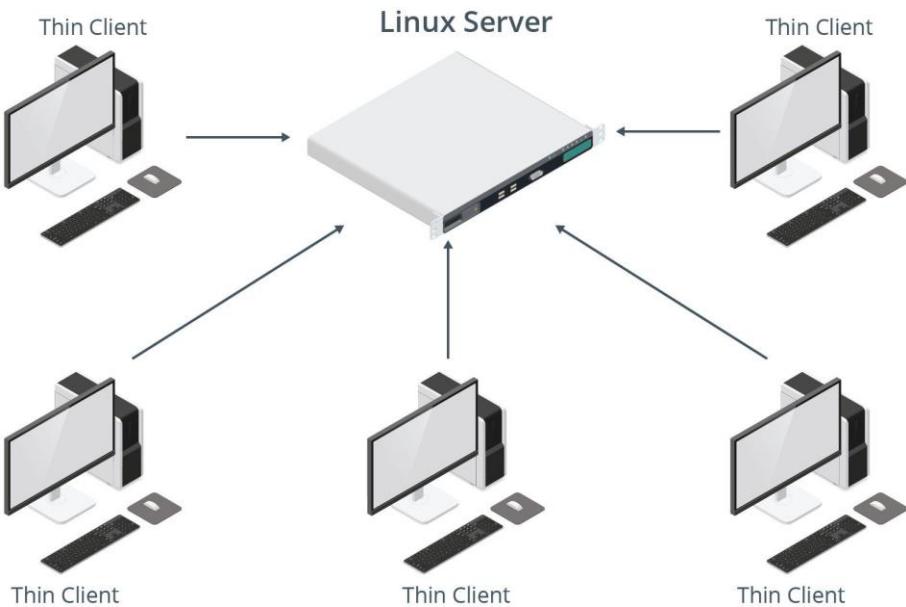
THE IMPORTANCE OF DEVICE DRIVERS

As you know, device drivers act as an interface between the operating system and hardware devices. Drivers are crucial for devices to function properly, or to even function at all within the Linux environment. While great strides have been made by both the Linux kernel developers and device manufacturers to write Linux-compatible drivers, there may still be compatibility issues with certain hardware. Ultimately, no matter what the device is that you're trying to use, you need to make sure it has proper driver support under Linux, and that your Linux system actually has those drivers.

Otherwise, you may be unable to use that device, or use it well.

THIN CLIENTS

A client device, typically referred to as a **thin client**, is any lightweight computing device that connects to a more powerful server for doing work. The server does most of the heavy lifting, including processing and storing data, while the thin client acts as little more than a user interface. This type of computing architecture centralizes operations, making it easier for administrators to manage groups of backend servers instead of workstations that are dispersed throughout the organization.



Thin clients connecting to a centralized, powerful server.

As you've seen, Linux has robust remote connection support, whether through a GUI with remote desktop apps, or through a terminal with SSH. A thin client will typically have fundamental I/O devices like a keyboard, mouse, and monitor connected to it. It may or may not be running Linux, and if it is, it will typically be a lightweight distribution with very few features available or permissible to the user. The server the client connects to will usually construct a virtual environment so that the user can work in a sandbox, segmented from other portions of the server or network. The server, like the client, may be running Linux or another operating system.

USB DEVICES

Universal Serial Bus (USB) is a peripheral interface technology that has become the *de facto* standard for connecting input devices, external storage devices, mobile devices, and more, to computers. USB also incorporates plug-and-play technologies that enable devices to self-configure as soon as a connection is made.

A wide range of USB device types are supported in Linux, including, but not limited to:

- Thumb drives
- External HDDs and SSDs
- Digital cameras
- Smartphones and tablets
- Printers and scanners
- Keyboards and mice
- Microphones and webcams
- Game controllers

USB STORAGE AND DEVICE ASSIGNMENT

Linux registers USB storage devices attached to the system in the format `/dev/sd#` in the same way as the SCSI/SATA naming convention.

WIRELESS DEVICES

Wireless devices transmit and receive signals over the air rather than through physical cables connected to ports. There are many different wireless networking protocols, each of which may have a specific or optimal application. Examples of common wireless networking protocols supported by Linux include:

- **Wi-Fi:** A technology used primarily in establishing a wireless local area connection (WLAN) in home and office environments. Common devices include wireless routers and access points for networking infrastructure; and mobile devices, desktop computers, Internet of Things (IoT) devices, and many more that act as Wi-Fi clients. Wi-Fi routers and access points may often run some flavor of Linux, whereas Linux-based clients like Android™ smartphones and desktop computers can connect to those infrastructure devices.
- **Bluetooth:** A technology used primarily for establishing a personal area network (PAN) in which devices communicate wirelessly within a few feet of each other. A common application is when wireless headsets or headphones are paired with a smartphone or other computing device, enabling the user to listen to audio without the need for cabling. As you've seen, Linux has Bluetooth® driver support and can act as either end of the pairing.
- **Near Field Communication (NFC):** A communications protocol used by mobile devices and peripherals that are either touching or only inches apart. NFC is often used to quickly share data from one device to another. As with Bluetooth, support for NFC on Android mobile devices is robust. There is also Linux support for some NFC adapters and discrete devices that connect to traditional computers.



Wireless devices in a WLAN.

VIDEO AND AUDIO DEVICES

Video and audio devices are I/O peripherals that are usually attached to client systems like desktops and laptops, or thin clients. Common video input peripherals include webcams, surveillance cameras, and digital cameras. Common video output peripherals include monitors, televisions, and projectors. Microphones are the most common audio input peripheral, but certain video input peripherals also capture audio. Likewise, monitors and televisions can usually output audio, but audio-only output devices like speakers and headphones are also popular.

When you connect a video/audio peripheral to a system running any OS, including Linux, you need to be mindful of the connection types it uses and what types are available on the system. Microphones and webcams commonly use USB, whereas USB is much less effective for streaming video/audio output in real-time. Monitors, for example, are more likely to use interfaces like HDMI and DisplayPort that can carry both video and audio signals with a high degree of real-time performance. Some monitors may use older interfaces like DVI and VGA that only carry video.

PRINTERS

Like most other operating systems, Linux provides support for printers. Support for a specific type of printer is dependent on whether or not there are available drivers for the Linux kernel to use, and how robust those drivers are. As driver support in the kernel has improved over the years, so too has support for many printers offered by most major vendors.

You can connect to a printer using one or more interfaces and methods. Most modern printers offer local connection support through a USB interface. However, in office environments especially, clients often connect to printers over a network. In this case, the printer may include a Wi-Fi adapter, an Ethernet adapter, or both, so that the printer is identifiable on the LAN. Multiple clients can therefore connect to and use the same printer at once. A Linux computer can even function as a print management server that interfaces with one or more physical printers.

NETWORK ADAPTERS

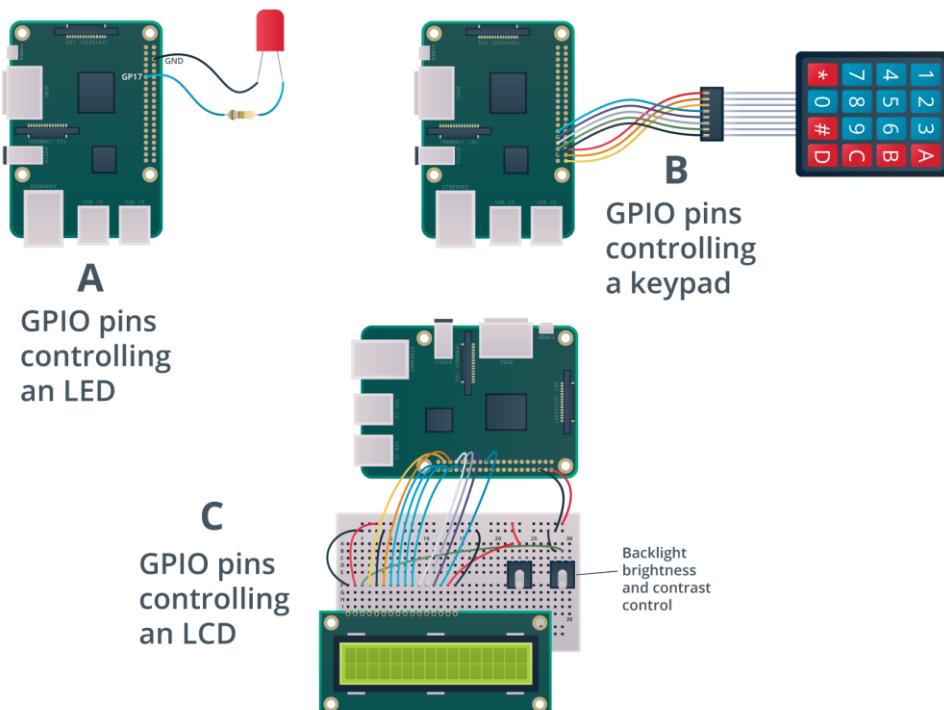
A **network adapter**, also known as a **network interface card (NIC)**, is a device that provides an interface with which hosts exchange data over a network. A network adapter is mandatory for any computing device that needs access to a network, whether a small LAN or a wider network like the Internet.

In many cases, network adapters are built into a computer's motherboard. However, some adapters can be added to the system on an expansion bus, or as an external peripheral that connects through an interface like USB. A system can have more than one adapter; this is especially common in servers and security systems like firewalls. In addition, each type of network connection protocol requires its own type of adapter. A Wi-Fi adapter sends and receives wireless signals in a WLAN, whereas an Ethernet adapter will include an Ethernet port that you can plug a cable into to connect to a LAN.

GPIO

General-purpose input/output (GPIO) refers to pins on a circuit board that have no designated purpose, but are controllable by the user at runtime. These pins send and receive digital signals and can be in an on or off state. For example, a pin designated as output can turn an LED light on or off; and a pin designated as input can itself be turned on or off from an external source like a light switch. In most cases, GPIO functionality is controlled programmatically through software. A developer might design their app to send or receive signals to and from the pins so that the app can interface with the outside world.

GPIO pins are commonly found on single-board microcontrollers like Arduino and Raspberry Pi devices. While these devices are often used by hobbyists and as a way to teach fundamental computing concepts, they also have applications in the IoT space. Specialized Linux distributions like Raspbian can be used to control GPIO functionality on single-board microcontrollers through various programming libraries.



GPIO pins on a Raspberry Pi device.

SATA

Serial AT Attachment (SATA) is a computer bus interface standard for attaching storage devices to traditional computers. In modern PCs, SATA has largely replaced earlier standards like Parallel ATA (PATA) and Integrated Drive Electronics (IDE) as one of the dominant standards in storage connection technology.

In the past, SATA was seen as a good solution for situations where capacity and cost were paramount. SATA supports multiple-terabyte drives and is relatively cheap to manufacture. However, its slower data transfer rate compared to alternative standards (6 gigabits per second) made it less suitable for enterprise environments, and it was most commonly used for backup purposes and in consumer storage. The most current revision of SATA (revision 3.2) combines SATA with another bus technology called PCI Express, which allows it to support a raw data rate of up to 16 Gb/s—finally rivaling its competitors.

SCSI

Small Computer System Interface (SCSI) is a computer bus interface for connecting peripheral devices to traditional computers. Whereas SATA is primarily used for attaching storage devices, SCSI can be used to attach other types of devices as well, such as DVD-ROM drives, printers, scanners, etc. However, its application in storage is much more common.

Traditionally, SCSI uses a parallel interface, which tends to cost more to manufacturer than a serial interface. However, in the past, SCSI provided high transfer rates and therefore became popular in enterprise storage arrays where speed was important. The **Serial Attached SCSI (SAS)** standard was developed to apply a serial interface to SCSI technology. SAS offers greater speeds than traditional SCSI—up to 24 Gb/s in the recent SAS-4—and supports higher-capacity drives. The serial interface also supports a more reliable data transfer rate. SAS has become the go-to technology for many enterprise storage environments.

HBA

A **host bus adapter (HBA)** is a hardware component that connects a host system to a storage device, like in a storage area network (SAN), in order to facilitate the input and output of data. They are to storage devices what network adapters are to networks. HBAs are commonly used with interface technologies like SATA and SCSI. The HBA might be built into the motherboard, or it might be a separate expansion card that attaches to the motherboard. In either case, you connect a storage device to an HBA with the requisite interface in order for the system to work with that storage device.

PCI

Peripheral Component Interconnect (PCI) is a connection interface standard that is primarily used as an expansion bus for attaching peripheral devices. The initial PCI specification has been largely superseded by the more recent **PCI Express (PCIe)**. PCIe supports greater transfer speeds, more reliable error detection, and is physically smaller than traditional PCI. In modern computers, PCIe is the dominant expansion bus technology.

When it comes to non-storage devices, there are many applications of PCIe. It's common for video cards to connect using this interface, as well as expansion cards that add more ports like USB and SATA for the system to use. In more recent years, PCIe has been used to connect SSDs that leverage the Non-Volatile Memory Express (NVMe) interface, which is specifically designed for integration with PCIe. NVMe SSDs are even faster than SSDs connected over SATA and SAS; they will likely become a prominent force in enterprise storage.

Activity 9-1

Identifying the Types of Linux Devices

SCENARIO

Part of your administrative duties at Develetech includes configuring and maintaining various hardware devices used in both client and server systems. Before you can apply your skills and knowledge in a Linux environment, you need to make sure you have a solid understanding of what the common devices are and how they connect to computers.

1. **True or false? Linux supports a unique set of device standards, and devices that work on other operating systems will usually not work with Linux.**
 True
 False
2. **What type of device is /dev/sdb?**
3. **Some users are concerned that their devices, whether USB, wireless, SCSI, etc., won't be compatible with Linux. What can you do to help ensure that these devices will work with Linux?**
4. **What is the purpose and function of a network adapter?**
 To translate wireless signals into physical data transmissions, and vice-versa.
 To provide an interface for connecting hosts to a network so that they can exchange data with one another.
 To establish a network segment that multiple computers can connect to in order to exchange data with one another.
 To forward transmitted data from one network to another.
5. **True or false? It is common for specialized Linux servers, such as firewalls, to contain more than one network adapter.**
 True
 False

6. **True or false? Printers can use a wide variety of interfaces, and can even be accessed over a network.**
 - True
 - False
7. **Which of the following hardware interfaces carries audio and video signals and is used to connect devices like monitors?**
 - USB
 - HDMI
 - Wi-Fi
 - VGA
8. **Which of the following wireless connection standards is primarily used to create a local area network (LAN) for home and office computing?**
 - NFC
 - Bluetooth
 - Wi-Fi
 - RFID
9. **Which of the following is a hardware component that connects a host system to a storage device to facilitate input and output of data?**
 - HBA
 - SATA
 - SCSI
 - GPIO
10. **Which of the following is a computer bus standard used by high- performance NVMe solid-state drives (SSDs)?**
 - SATA
 - SCSI
 - PCIe
 - SAS

11. Which of the following hardware connection technologies is primarily used in enterprise storage because of its fast speeds and high reliability?

- SCSI
 - SAS
 - PCI
 - SATA
-

Topic B

Configure Devices



EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

Now you're ready to use Linux tools to configure devices. In this topic, you'll use some common services that enable you to customize how the system interacts with certain hardware.

DEVICE FILE LOCATIONS

Device files represent information about hardware devices, and in some cases, define settings about those devices that you can customize. These files are located in several different directories and subdirectories, many of which you've seen before.

Location	Description
/proc/	Contains various files that represent system information reported by the kernel. In particular, the /proc/devices file contains a list of all device drivers that the kernel is currently running, separated by character and block storage devices.
/sys/	A virtual file system similar to /proc/ but that focuses more on creating a hierarchical view of device information. In particular, the /sys/devices/ subdirectory includes files that expose details about specific devices.
/dev/	Contains device driver files that enable the system and users to access devices themselves. For example, you can mount a block storage device by referencing it from /dev/ , like /dev/sda1 . Likewise, /dev/mapper/ contains logical volumes, encrypted volumes, and other devices that are managed by device mapper.
/etc/	Includes configuration files for many components, including components that interface with devices. For example, the /etc/X11/ subdirectory contains configuration files for input and output devices that might impact the X.Org Server environment, such as keyboards, mice, and monitors.

HOTPLUGGABLE DEVICES

A **hotpluggable device** can be physically added or removed from the system without requiring a reboot in order to use that device. Hotpluggable devices are detected by the system as they are plugged in, whereas coldpluggable devices, such as RAM modules, CPUs, and some internal storage devices, are not sensed when connected to a running system; they need a complete reboot of the system to function. In fact, for internal devices like RAM modules, it is highly recommended that the system is powered off before attempting to plug the device in.

Modern Linux distributions support hotplugging for many standard bus types, particular for USB, FireWire, SATA, and other related technologies. Even expansion bus technology like PCIe can support hotplugging.

udev

The device manager **udev** manages the automatic detection and configuration of hardware devices. A function of systemd, udev is an integral part of the kernel that is initialized during boot time. The udev utility handles module loading for both coldpluggable and hotpluggable devices. It loads the modules for coldpluggable devices when the system is booted. The modules for hotpluggable devices are loaded by udev dynamically during system run time.

udev RULES

The **/etc/udev/rules.d/** directory is used to configure rules for how udev functions. You can create files in this directory that tell udev to configure a device in a certain way or run a certain command when a device is plugged in. For example, you might want to create a symbolic link to a specific device every time it is plugged in; that way, you can always refer to this device in the same way, rather than relying on the unpredictable and non-descriptive **/dev/** naming scheme like **/dev/sda1**, **/dev/sdb2**, etc. In your rules file, you'd need to refer to the device by attributes that are unique to that device, like its vendor and product IDs. For example, the following line in a rules file will create a symbolic link to a specific USB thumb drive when it is plugged in:

```
KERNEL=="sd*", ATTRS{idVendor}=="334455",
ATTRS{idProduct}=="667788", SYMLINK+="myusb"
```

Similar to writing rules for GRUB, you name this rule file in the format **##-name.rules** where **##** determines its order in being executed by udev.

ADDITIONAL udev RULES DIRECTORIES

There are actually several directories that are used to configure udev rules.

The **/etc/udev/rules.d/** directory mentioned previously is primarily used for local administration of udev. An administrator applies their own customizations to this directory so that udev behaves in accordance with the administrator's preferences and/or the organization's business needs. As a result, the files in this directory are loaded with the highest priority.

The **/usr/lib/udev/rules.d/** directory also contains udev rules. However, these rules are generated by the system, and you should refrain from editing them. Rules in this directory are low priority, so a custom rule named **60-keyboard.rules** in **/etc/udev/rules.d/** will supersede the default **60-keyboard.rules** file in the **/usr/lib/udev/rules.d/** path.

Rules files can also be placed in the **/run/udev/rules.d/** directory. These also take precedence over the system rules path. Rules in this path are volatile, meaning that they will apply at runtime but will be lost in the event of a system reboot. Volatile rules can be useful when you need to temporarily override a system rule without actually making the change persist.

THE udevadm COMMAND

The **udevadm** command is used to manage udev. It takes various subcommands, each of which performs a certain task to modify the behavior of the **systemd-**

udev daemon and related components. Some of these subcommands are described in the following table.

Subcommand	Used To
info	Retrieve device information stored in the udev database, as well as detailed device attributes from the /sys/ file system. For example, you can view a device's vendor ID, product ID, serial number, and much more.
control	Modify the running state of udev. For example, providing the --reload-rules option will ensure that udev is reading from any new rules files you've added.
trigger	Execute rules that apply to any device that is currently plugged in. You can also specify an action using the -C option, such as add , remove , or change . As the names imply, these will trigger events where a device is added, removed, or changed in the running kernel.
monitor	Watch for events sent by the kernel or by a udev rule.
test	Simulate a udev event running for a device, with results on output.



The diagram shows a callout box labeled "Device attributes" pointing to a terminal window. The terminal window displays the output of the command "root@server01 ~]# udevadm info /dev/sda1". The output lists various device attributes, including P:, N:, S:, E:, and E: entries for DEVNAME, DEVPATH, DEVTYPE, ID_ATA, ID_ATA_DOWNLOAD_MICROCODE, and ID_ATA_FEATURE_SET_APM.

```

root@server01 ~]# udevadm info /dev/sda1
P: /devices/pci0000:00/0000:00:1f.2/ata1/host0/target0:0:0/0:0:0:0/block/sda/sda1
N: sda1
S: disk/by-id/ata-ST500LM000-SSHD-8GB_W763QY42-part1
S: disk/by-id/wwn-0x5000c5007d80171d-part1
S: disk/by-partlabel/EFI\x20System\x20Partition
S: disk/by-partuuid/d73a6d27-b049-497e-8f53-c51c7b1a614c
S: disk/by-path/pci-0000:00:1f.2-ata-1.0-part1
S: disk/by-uuid/E434-7404
E: DEVLINKS=/dev/disk/by-id/ata-ST500LM000-SSHD-8GB_W763QY42-part1 /dev/disk/by-id/wwn-0x5000c5007d80171d-part1 /dev/disk/by-partlabel/EFI\x20System\x20Partition /dev/disk/by-partuuid/d73a6d27-b049-497e-8f53-c51c7b1a614c /dev/disk/by-path/pci-0000:00:1f.2-at-a-1.0-part1 /dev/disk/by-uuid/E434-7404
E: DEVNAME=/dev/sda1
E: DEVPATH=/devices/pci0000:00/0000:00:1f.2/ata1/host0/target0:0:0/0:0:0:0/block/sda/sda1
E: DEVTYPE=partition
E: ID_ATA=1
E: ID_ATA_DOWNLOAD_MICROCODE=1
E: ID_ATA FEATURE_SET_APM=1

```

Displaying device attributes and other information.

SYNTAX

The syntax of the **udevadm** command is **udevadm [options] [subcommand] [arguments]**

PRINTING SOFTWARE

Printers are typically bundled with software utilities that enable you to configure settings for the printer. These utilities may target a specific operating system, so you need to confirm whether or not they were designed to run on Linux. Even if you cannot run the manufacturer's software utilities, you may still be able to work with the printer through a Linux utility. Major vendors will usually provide the most up-to-date drivers for download off their websites.

CUPS

CUPS is a print management system for Linux that enables a computer to function as a print server. A system running CUPS is a host that can initiate print jobs from client systems. These jobs are then processed and sent to the appropriate printer. The main advantage of CUPS is that it can process different data formats on the same print server.

CUPS is designed for scheduling print jobs, processing administrative commands, and providing printer status information to local and remote programs. CUPS provides a web-based interface for configuring the service. Changes made through this interface modify the `/etc/cups/cupsd.conf` and `/etc/cups/cups-files.conf` files.



Note: CUPS used to stand for Common Unix Printing System and was developed by Apple.

The screenshot shows the CUPS web interface with the following sections:

- Printers:** Buttons for Add Printer, Find New Printers, and Manage Printers.
- Classes:** Buttons for Add Class and Manage Classes.
- Jobs:** Button for Manage Jobs.
- Server:** Buttons for Edit Configuration File, View Access Log, View Error Log, and View Page Log. Below this is a section titled "Server Settings:" with an "Advanced" link and several checkboxes:
 - Share printers connected to this system
 - Allow printing from the Internet
 - Allow remote administration
 - Use Kerberos authentication ([FAQ](#))
 - Allow users to cancel any job (not just their own)
 - Save debugging information for troubleshooting

The CUPS web interface.

THE **Ipr** COMMAND

The **Ipr** command submits files for printing. Files supplied at the command-line are sent to the specified printer or to the print queue if the printer is busy. Without specifying the printer to use, the command will send the print job to the default printer, which you can configure with CUPS. The **Ipr** command reads the print file from standard input if no files are supplied at the command-line.

SYNTAX

The syntax of the **Ipr** command is **Ipr [options] [file names]**

Ipr COMMAND OPTIONS

The **Ipr** command options are described in the following table.

Option	Used To
-E	Force encryption when connecting to the server.
-P {destination}	Send the print job to the destination printer specified.
-# {copies}	Set the number of copies to print from 1 to 100.
-T {name}	Set the job name.

Option	Used To
-l	Specify that the print file is already formatted and should be sent to the destination without being filtered.
-o {option}	Set a job option, like printing in landscape mode, scaling the printed output, printing double-sided, etc. Job options vary depending on the printer.
-p	Print the specified files with a shaded header that includes the date, time, job name, and page number.
-r	Specify that the printed files should be deleted after printing.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 9-2

Discussing Configuring Devices

SCENARIO

Answer the following questions to check your understanding of the topic.

-
1. **What task can you perform to be able to refer to a device with the same custom name every time it is plugged in?**

 2. **What is the significance of udev?**

 3. **How can you start the CUPS service?**

 4. **How do you send the file students.txt to the HPLJ5 printer?**

 5. **There are two device types recognized by the udev device management system: coldpluggable and hotpluggable. When does the system load modules for each device type?**
-

Activity 9-3

(Optional) Using udev to Manage a Hotpluggable Device

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have a USB thumb drive.

SCENARIO

One of your colleagues uses a Linux system to prep OS installation media on a USB thumb drive. However, every time she plugs her USB drive into the system, it automatically assigns the drive a non-descriptive name like `/dev/sdb`. She'd like to be able to reference the drive with a more helpful name every time she plugs it in. So, you'll use `udev` to create a symbolic link to the drive based on its unique attributes.

1. Gather information unique to your USB thumb drive.

- a) Plug your USB thumb drive into your computer.
- b) In a terminal, enter `lsblk` and note the device name of your USB drive:

It'll likely be something similar to `sdb`. Use the column that displays maximum storage size if you need help identifying the device.

- c) Enter `udevadm info /dev/<device name>` where `<device name>` is the name of the device you just noted.

For example: `udevadm info /dev/sdb`

- d) Verify that you can see various identifying information about the device.
- e) Enter `udevadm info -a /dev/<device name> | less` to retrieve device attributes.
- f) Enter `/serial` to search for the device's serial number, then make note of it:

- g) Look at the rest of the attributes of this parent device.

The `idVendor` and `idProduct` attributes are particularly useful because they tend to be unique to each device.

- h) Note the `idVendor` and `idProduct` attributes:

`idVendor:` _____

`idProduct:` _____

- i) Keep this window open.

2. Create a `udev` rule that will automatically link a custom device name to the USB drive when it's plugged in.

- a) Right-click within the terminal and select **Open Terminal** to open a new window.
- b) In the new terminal window, change to the `/etc/udev/rules.d` directory.

- c) Using **Sudo**, create a text file named **10-local.rules** with the text editor of your choice.
- d) In this text file, enter the following all on one line:

```
KERNEL=="<device name>", ATTRS{serial}=="<value>",
ATTRS{idVendor}=="<value>", ATTRS{idProduct}=="<value>",
SYMLINK+="install"
```

Fill in the replacement values with the values identified in your other terminal. Make sure the values are from the same parent device that you identified as having the serial number. Also, **<device name>** should be in the format **Sdb** and *not /dev/sdb*

- e) Save and close the file.
- 3.** Ensure the new rule is reloaded and triggered.

- a) Enter **Sudo udevadm control --reload-rules**

udev usually reloads rules automatically, but you can issue this command to make sure.

- b) Enter **udevadm trigger**

This will trigger the rules for any devices currently connected. Otherwise, you'll need to disconnect and then reconnect the device for the rule to trigger, which you'll do in the next step.

- 4.** Test the rule to see if it successfully creates the symbolic link.

- a) From the desktop menu, select **Applications**→**Favorites**→**Files**.

- b) On the navigation pane, select the **Unmount** button next to your USB thumb drive.

- c) Unplug your USB drive from the computer.

- d) Plug your USB drive back in.

- e) At a terminal, enter **lsblk /dev/install**

- f) Verify that the USB drive is displayed, indicating that the symbolic link was activated automatically when you connected the drive.

- g) Unmount and unplug your USB drive.

- h) Close the file browser window.

Activity 9-4

Configuring a Virtual (PDF) Printer

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have a `~/policies/aup_v2.txt` file.

SCENARIO

HR wants to distribute the acceptable use policy (AUP) to employees at Develetech in both hardcopy and electronic form. Right now, you don't have an actual printer connected to your Linux system, but you can still print the AUP text file to a PDF, which is more suitable than a raw text file for distribution purposes. Before you can create the PDF, you'll need to set up a virtual printer.

1. Examine the current list of printers.
 - a) From the desktop menu, select **Applications**→**System Tools**→**Settings**.
 - b) From the navigation menu, select **Devices**.
 - c) Select **Printers**.
 - d) Verify that no printers are currently listed.
 - e) Keep this window open.
2. Install the **Cups-PDF** package.
 - a) In a terminal, enter `sudo yum -y install epel-release cups-pdf`

Note: You may need to issue this command twice in order to successfully download the package.

3. Make the **Cups-PDF** printer your default printer.
 - a) Return to the **Settings** window and verify that **Cups-PDF** is listed in the printers list.



- b) Select **Unlock**, then enter the root password.
- c) Next to the **Cups-PDF** printer, select the options gear icon, then select **Use Printer by Default**.
- d) Close the **Settings** window.

4. Print a text file to PDF.

- a) Open a terminal window and verify that you are in your home directory.
 - b) Enter **lpr policies/aup_v2.txt**
 - c) Verify that a printing notification pops up from the desktop.
 - d) On the desktop, double-click **aup_v2.pdf** to open it.
 - e) Close the PDF viewer when you're done.
-

Topic C

Monitor Devices



EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

Now that your device configurations are in place, you'll need to monitor those devices to ensure they are recognized by the system and performing as expected. So, in this topic, you'll gather information about connected hardware.

THE `lsdev` COMMAND

The `lsdev` command displays various information about a system's hardware as reported by the kernel. It compiles this information from three files in the `/proc/` directory:

- **`/proc/interrupts`** —This file lists each logical CPU core and its associated interrupt requests (IRQ). An IRQ is a signal sent by a device to the processor so that the processor can stop what it is doing and handle some task that the hardware needs to perform, like pressing a keystroke or moving the mouse. There are multiple IRQ addresses that signals can be sent along, and for each address, this file lists how many signals were sent to each CPU core along that address. It also names the hardware device that is mapped to each IRQ address.
- **`/proc/ioports`** —This file lists I/O ports and the hardware devices that are mapped to them.
- **`/proc/dma`** —This file lists all Industry Standard Architecture (ISA) direct memory access (DMA) channels on the system. ISA DMA is a hardware controller that typically supports legacy technology like floppy disks.



Note: The `lsdev` command is more common on Debian-based distributions and is available from the `procinfo` package.

THE `lsub` COMMAND

The `lsub` command is used to display information about devices that are connected to the system's USB buses. This command scans the `/dev/bus/usb/` directory for information. By default, the command will print the number of the bus and the connected device, the ID of the device, and the name of the vendor and product matching that device.

You can use the `-V` flag to see detailed information about each device, similar to using the `udevadm info` command. You can also filter results by bus (`-S`) and by vendor/product (`-d`).

```
root@server01 ~]# lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 03eb:8a1c Atmel Corp.
Bus 001 Device 006: ID 0cf3:3004 Qualcomm Atheros Communications AR3012 Bluetooth 4.0
Bus 001 Device 003: ID 598c:055d Acer, Inc
Bus 001 Device 002: ID 04f2:0939 Chicony Electronics Co., Ltd
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Listing USB device information.

SYNTA X

The syntax of the **lsusb** command is **lsusb [options]**

THE lspci COMMAND

The **lspci** command is used to display information about devices that are connected to the system's PCI buses. By default, the output will list the logical slot address (typically in the format **Bus:Device.Function**), the device's class (such as network controller, storage controller, input device, or bridge device), the vendor name, and the device name. Like **lsusb**, **lspci** offers a verbose mode for more detailed information about each device. For example, you can use verbose mode to identify the physical slot in which an adapter is installed.

```
root@server01 ~]# lspci
00:00.0 Host bridge: Intel Corporation Haswell-ULT DRAM Controller (rev 0b)
00:02.0 VGA compatible controller: Intel Corporation Haswell-ULT Integrated Graphics Controller (rev 0b)
00:03.0 Audio device: Intel Corporation Haswell-ULT HD Audio Controller (rev 0b)
00:14.0 USB controller: Intel Corporation 8 Series USB xHCI HC (rev 04)
00:16.0 Communication controller: Intel Corporation 8 Series HECI #0 (rev 04)
00:1b.0 Audio device: Intel Corporation 8 Series HD Audio Controller (rev 04)
00:1c.0 PCI bridge: Intel Corporation 8 Series PCI Express Root Port 1 (rev e4)
00:1c.2 PCI bridge: Intel Corporation 8 Series PCI Express Root Port 3 (rev e4)
00:1c.3 PCI bridge: Intel Corporation 8 Series PCI Express Root Port 4 (rev e4)
00:1f.0 ISA bridge: Intel Corporation 8 Series LPC Controller (rev 04)
00:1f.2 SATA controller: Intel Corporation 8 Series SATA Controller 1 [AHCI mode] (rev 04)
00:1f.3 SMBus: Intel Corporation 8 Series SMBus Controller (rev 04)
02:00.0 Network controller: Qualcomm Atheros QCA9565 / AR9565 Wireless Network Adapter (rev 01)
03:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Exp
```

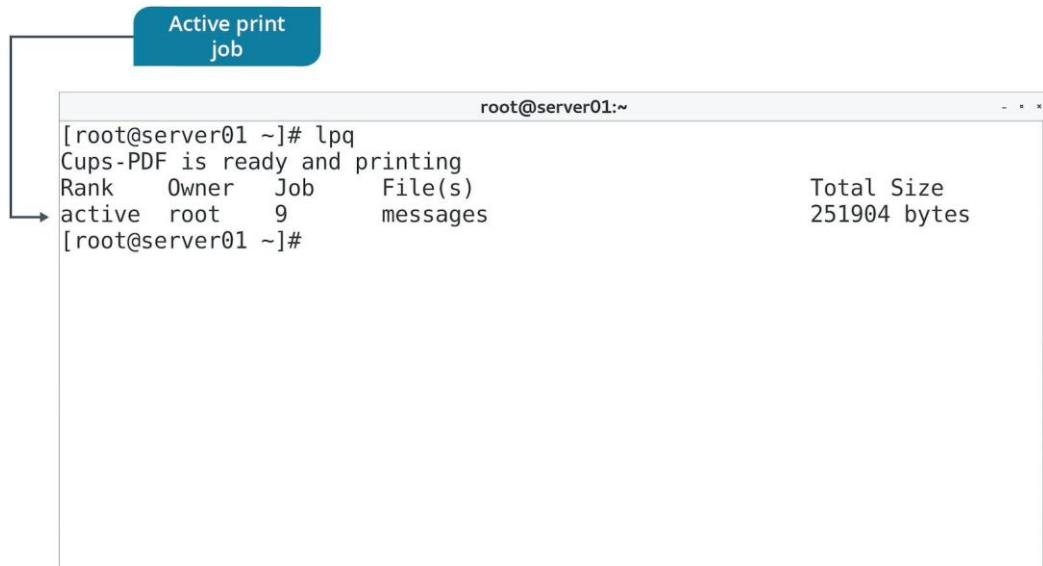
Listing PCI device information.

SYNTA X

The syntax of the **lspci** command is **lspci [options]**

THE **lpq** COMMAND

The **lpq** command shows the status of the printer queue. By default, it will report each print job's rank in the queue, who owns the job, the job number, the files in the job, and the size of the job. You can also have the report update every number of seconds that you specify with the **+interval** option, until the queue is empty. If you don't specify the printer to monitor, the **lpq** command will monitor the default printer.



```
root@server01:~# lpq
Cups-PDF is ready and printing
Rank   Owner   Job      File(s)          Total Size
active  root     9       messages        251904 bytes
[root@server01 ~]#
```

Listing the printer queue.

SYNTAX

The syntax of the **lpq** command is **lpq [options]**

ADDITIONAL DEVICE MONITORING TOOLS

Some tools you've already used thus far can also be useful in monitoring hardware devices. For example, you can use **lsblk** to identify block storage devices connected to the system. The output of this command can help you ensure that your storage devices are recognized and that they are correctly partitioned and mounted.

Another example is the **dmesg** command. Recall that this prints all messages sent to the kernel's message buffer after system boot, including messages sent by device drivers. If hardware devices encounter errors in operation or are unable to load the expected modules into the kernel, the output of **dmesg** might indicate so. Use this output to monitor for issues related to device drivers and the underlying hardware they control.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 9-5

Discussing Monitoring Devices

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **You want to check the print queue before you attempt to reboot the system. How do you check the print queue for the non-default printer, HPLJ5?**
2. **What does the `lpq -a` command display?**
3. **What does the `lsusb` command display?**
4. **Which command can you use to display information about all block devices and how can you find out more information about each device in the list?**
5. **From where does the `lsdev` command pull its displayed information?**

Activity 9-6

Monitoring Devices

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

As part of your routine system administration tasks, you need to track the devices used on all the computers on the network and maintain a list of hardware resources that are in use.

1. Monitor any PCI devices that are connected to the system.
 - a) In a terminal window, enter **lspci**
 - b) Examine the brief list of hardware devices that are connected to PCI buses.
Each line lists the PCI slot a device is connected to, as well as the vendor and product model of the device.
 - c) Enter **sudo lspci -v** to view more information about each device.
 - d) Verify that you can see attributes of each device, such as:
 - Its vendor and product model.
 - Its I/O ports.
 - Its various capabilities.
 - The kernel modules and drivers it uses.

2. Monitor any USB devices that are connected to the system.
 - a) Enter **lsusb**
 - b) Examine the brief list of hardware devices that are connected to USB buses.
Each line lists the USB bus a device is connected to, as well as the vendor and product model of the device.
 - c) Enter **sudo lsusb -v** to view more information about each device.
 - d) Verify that you can see attributes of each device, such as:
 - Its vendor and product model.
 - The specification version it supports (**bcdUSB**).
 - The maximum power the device can drain from the bus (**bMaxPower**).
 - The class of device supported, e.g., mass storage vs. communications (**bInterfaceClass**).

3. Monitor a print job.
 - a) Enter **base64 /dev/urandom | head -c 10000000 > printfile.txt**

Note: There are 7 zeros.



This will create a text file 10 MB in size using random data encoded in readable text (Base64). This file is large enough to take some time to print, so you'll have a chance to monitor it in the queue.

- b) Enter **lpr printfile.txt** to start a print job.
 - c) Enter **lpq** and note that you can see the print job in the queue, including information such as:
 - Its rank (status).
 - Who owns the print job.
 - The number of the job.
 - What file(s) are being printed.
 - The total size of the request.
 - d) Allow the print job to finish.
-

Topic D

Troubleshoot Hardware Issues



EXAM OBJECTIVES COVERED

2.7 Explain the use and operation of Linux devices.

4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

As you might expect, hardware devices are just as susceptible to problems as any other Linux component. In this topic, you'll look at some of the common symptoms that indicate device issues, as well as some suggested solutions for dealing with those issues.

COMMON HARDWARE ISSUES

Problems can affect a wide array of different hardware devices. Missing or poorly configured drivers are a common source of these problems, as is user space software that is incompatible with certain hardware. However, there are many other potential sources. Likewise, there may be many potential solutions that differ based on the type of component you're troubleshooting. In general, hardware issues can be categorized as follows:

- Keyboard mapping issues.
- Communications port issues.
- Printer issues.
- Memory issues.
- Video issues.
- Storage adapter issues.

KEYBOARD MAPPING ISSUES

Certain keyboard keys, when pressed, may produce an unexpected character on the screen, or no character at all. This is the most common and overt symptom of a keyboard mapping issue. The most likely cause of these issues is that the system has configured the wrong keyboard layout and/or the wrong language.

To address the problem, make sure you can correctly identify the layout of the physical keyboard—not just its overall design type (e.g., QWERTY vs. Dvorak), but its specific regional layout. Even standard QWERTY keyboards designed with the same language in mind don't always have the same layout; for example, American English keyboards contain keys in different spots than British English keyboards. Once you've identified the physical layout of your keyboard, use **localectl status** to verify the layout that the system is using. If it's not correct, list the available keymaps, identify the correct one, then set it on the system. You may also need to adjust the system language to ensure one-to-one accuracy.

ADDRESSING ISSUES WITH A REMOTE TERMINAL

If you are accessing a Linux system remotely, your remote terminal client may have some options for addressing keyboard mapping issues. SSH clients like PuTTY enable you to change the effects that certain keystrokes have on the environment. For example, you can configure the **Backspace** character to move one character to the left

without deleting the character. This can be helpful in certain applications that don't handle the default behavior well.

COMMUNICATIONS PORT ISSUES

Communications ports, like USB, may fail to recognize an attached device, or the connection to that device may be unreliable. Your first step should be to ensure that the device is correctly slotted into the port, and that any physical cables are not loose or damaged. Also make sure that power is being supplied to the bus adapter. If this doesn't fix the issue, then ensure that any necessary drivers are installed and loaded into the kernel, depending on the type of interface you're using. Also ensure your device supports the correct version of the bus interface. For example, each version of USB has a maximum data throughput value, and older versions may not meet your performance expectations.

Certain devices, when connected to a serial port, will request a console interface with the Linux operating system. Linux will typically assign the port an interface at `/dev/ttYS#` where # is the number of the console (starting with 0). In some cases, you may need to ensure that the connected device is configured to automatically use one of these consoles. Also, by default, only the root user is granted access to these serial consoles. You may need to change the permissions on the relevant serial console using `chmod` to ensure that other users can work with it.

PRINTER ISSUES

Printers are a very common source of issues. In many cases, the printer itself will be at fault:

- It may be out of ink or paper.
- There may be a paper jam.
- The mechanical components may be damaged or misaligned.
- And many more issues.

For these issues, consult the printer's help manual and/or the manufacturer's website.

In other cases, however, you may be able to troubleshoot issues from your Linux client or server. As always, ensure your specific printer is supported by Linux-compatible drivers, and that those drivers are loaded. If you're trying to connect to your printer over a network but can't, use network diagnostic tools like `ping` to ensure that your printer is identifiable on the network and that it can be reached.

If you're using Linux as a print server in an office environment, the printer may become sluggish or unresponsive if multiple users are trying to print to it. Use `lpq` to check the status of print jobs; if any jobs are too large, or there are too many in the queue, you can use the `lprm` command to stop a job with the job number you provide. For example, `lprm 4` will remove job 4 from the queue. This will help clear up the queue and lighten the load on the printer.

MEMORY ISSUES

From a software perspective, memory can "leak" when a process fails to free up allocated memory when it is no longer needed. The total available memory on the system is quickly exhausted. This can lead to general performance degradation and system instability because other software is unable to access the memory it needs. In these cases, you can use memory monitoring tools like `free` as well as process monitoring tools like `top` to identify the problem, then deal with the offending process (e.g., by killing it).

However, some memory issues indicate a fault in the physical RAM modules or the motherboard that RAM is slotted into. Like other operating systems, Linux has ways of detecting these faults during operation. For example, system logs that record a "Machine Check Exception" error message usually indicate an issue with RAM. The **mcelog** command can retrieve and print these error messages for easier analysis. If the messages contain error-correcting code (ECC) errors, one of the memory modules has probably failed.

To confirm RAM module failure, you can use utilities like MemTest, MemTest86+, and **memtester** to perform a stress test on all RAM modules for several hours, reporting any errors that are encountered.

VIDEO ISSUES

Common video-related issues include:

- Consistent or intermittent blank screens.
- Incorrectly displayed colors.
- Multiple monitors not being detected.
- Sluggish performance in video-intensive applications.
- And more.

Some of these issues can be addressed by ensuring that monitors and other display devices are properly connected and are compatible with the system and user software.

When it comes to performance of video-intensive applications, GPU driver support is one of the biggest hurdles. Linux has historically been less well-supported by GPU manufacturers than Windows, particularly with the two main GPU vendors: AMD and Nvidia. However, support from both vendors has improved in recent times. It's crucial to have the latest drivers in order to ensure optimal video performance. These drivers are made available for download from the vendor's website.

STORAGE ADAPTER ISSUES

There are several possible indicators of a faulty bus adapter, including:

- Poor data transfer speeds.
- Less total space available than expected.
- Excessive read/write errors.
- Inability to read/write at all.
- The system cannot detect devices at all.
- And more.

The problem might be with the physical HBA itself, or it might be with the interface that the HBA uses, such as SCSI or SATA. The first step is to ensure that the HBA is powered. Then, you need to ensure that the storage device you're connecting to the HBA uses the appropriate interface. Even though different generations of SCSI and SATA are usually backward compatible, if the bus interface uses older technology than the drive, then the drive will be limited to that older specification. And, of course, ensure that all devices are properly slotted and all cables are connected and damage-free.

In some cases, if the system doesn't recognize a new SCSI device, you may need to rescan the SCSI bus it's attached to. The following command rescans a specific SCSI bus:

```
echo " - - - > /sys/class/scsi_host/host#/scan
```

The **#** represents the number of the bus you're trying to scan. The hyphens in the **echo** statement are wildcards for SCSI controller, SCSI channel, and logical unit number (LUN), respectively. This will prompt the system to scan for new devices on this

bus and add any that are detected. However, this process can be disruptive, so it should only be used when necessary.

RAID TROUBLESHOOTING

As you've seen, the `mdadm` command is used to manage RAID arrays. The `-F` option activates monitor mode, enabling you to identify missing or failed drives in an array. You can then use some of the command's other modes to rebuild an array after you've removed the faulty drive.

Some other useful `mdadm` options for troubleshooting RAID issues are as follows:

- **-f**—Mark a specified device as faulty to prepare it for removal from the array.
- **-r**—Remove the specified device from the array. Use the keyword **failed** to specify that all devices marked as faulty should be removed.
- **--re-add**—Add a removed device back to the array for the purpose of recovering data stored on the device.
- **-a**—Add a device to the array as a hot-spare. If the array is degraded, it will rebuild data on that spare. This behavior only applies to devices that are unable to be re-added or were never part of the array to begin with.

THE `lshw` COMMAND

The `lshw` command lists each detected hardware component on the system and provides details about each device. The command pulls information from many different files in multiple device file locations like `/proc/` and outputs in a hierarchical format.

Information that `lshw` outputs includes the vendor, product name, capacity, speed, and many other attributes of the motherboard, CPU, RAM modules, peripheral devices, storage devices, and so on.

Like other commands and files that retrieve device information, you can use `lshw` to identify whether or not a device is recognized by the kernel, as well as to review a device's capabilities and characteristics.



Note: Depending on the version, the `lshw` command may not detect FireWire devices.
Check its man page to verify.

```
[root@server01 ~]# lshw -c input
*-usb:0
  description: Mouse
  product: USB Optical Mouse
  vendor: Chicony Electronics Co., Ltd
  physical id: 1
  bus info: usb@1:1
  version: 2.00
  capabilities: usb-2.00
  configuration: driver=usbhid maxpower=98mA speed=2Mbit/s
*-usb:3
  description: Human interface device
  product: Atmel maXTouch Digitizer
  vendor: Atmel
  physical id: 8
  bus info: usb@1:8
  version: 10.99
```

Listing information about hardware devices.

SYNTAX

The syntax of the `lshw` command is `lshw [options]`

DEVICE CLASSES

The output of `lshw` groups devices into one of several classes. You can filter the total results by specifying a class with the `-C` option. For example, issuing `lshw -C network` will only output details about network-class devices. To see a list of classes currently in use on your system, enter `lshw -short | sort -k2` to generate a non-detailed list of devices, sorted by the class column.

THE dmidecode COMMAND

The `dmidecode` command dumps the system's Desktop Management Interface (DMI) table and presents it in a human-readable format. The DMI table is an industry standard for tracking information about hardware components. It separates components into types, with each type given a number—for example, type 4 is a processor, type 39 is a power supply, etc. Like similar commands, you can use `dmidecode` to verify connected devices and whether or not they support certain features. However, the authors of `dmidecode` caution that the information in DMI tables is, more often than not, "inaccurate, incomplete, or simply wrong." Therefore, don't rely on DMI tables as the sole source of hardware information.

SYNTAX

The syntax of the `dmidecode` command is `dmidecode [options]`

ABRT

The **Automatic Bug Reporting Tool (ABRT)** is a utility, typically used on Fedora- and RHEL-based distros, that analyzes and reports on problems detected during system runtime. ABRT collects data like memory dumps from crashed applications to help administrators diagnose and troubleshoot issues. It can also report on problems with various devices, such as MCEs that typically indicate hardware failure.

ABRT can redirect problem data to many different destinations, including public issue trackers like Bugzilla and support sites like Red Hat Technical Support (RHTSupport). Or, it can simply write to a local or remote file in a standard format. The default location for problem data is in `/var/spool/abrt/` with timestamped subdirectories for each problem detected.

ABRT UTILITIES

ABRT runs as the `abrtd` daemon and can be configured using `abrt-cli` or `abrt-gui`, depending on your system (and your own preference). You can use both utilities to list problem data, view details about problem data, analyze and report on problem data, and remove unnecessary reports.

GUIDELINES FOR TROUBLESHOOTING HARDWARE ISSUES

Use the following guidelines when troubleshooting hardware issues.

TROUBLESHOOT HARDWARE ISSUES

When troubleshooting hardware issues:

- Ensure that hardware devices are supported through robust drivers.
- Ensure that the necessary drivers are installed and loaded in the kernel.
- Ensure that hardware devices are compatible with the Linux software that controls, manages, or interfaces with them.
- Verify that the system has the correct keyboard layout and language set.
- Verify that a network-enabled printer is identifiable on the network.
- Stop large or numerous print jobs with the **lprm** command.
- Check the **mcelog** for memory errors.
- Run a utility like **memtester** to stress test RAM modules.
- Download the latest GPU drivers from the vendor's website.
- Ensure storage and peripheral devices are properly slotted into the correct buses.
- Ensure connected cables are not loose or damaged.
- Use a command like **Ishw** to identify connected hardware.
- Be aware that **dmidecode** may produce inaccurate results.
- Review crash data compiled by the ABRT utility.

Activity 9-7

Troubleshooting Hardware Issues

SCENARIO

The Linux systems at Develetech fulfill a wide variety of purposes, and as such, must interface with many different kinds of hardware. Part of your responsibilities includes troubleshooting the inevitable issues that arise with hardware devices.

1. **What is a common source of problems with keyboards, printers, and storage adapters?**

2. **What is the term used to describe the situation where a process fails to free up allocated memory when it is no longer needed?**

3. **What are two common hardware-related issues that are not a result of faulty drivers or insufficient memory?**

4. **Which of the following commands is an often unreliable way of querying the system for attached hardware components?**
 - lshw
 - vmstat
 - free
 - dmidecode

5. **Which of the following connection protocols may lshw *not* detect?**
 - USB
 - FireWire
 - SCSI
 - PCI

6. A user is complaining that their USB mouse is unresponsive. Which of the following commands can you use to quickly pull up information about the device, including its vendor, product model, and drivers?
- lsht -c input
 - lsht -c mouse
 - dmidecode -t input
 - dmidecode -t mouse
7. When a user presses the Shift+2 on their keyboard, they expect the @ symbol to be entered, but instead, the # symbol is. What can you do to troubleshoot this issue?
8. Several users have been issuing multiple print jobs to the printer, causing it to become backed up and unresponsive. You enter the lpq command to query the print jobs, and notice that one job is very large and likely causing the bottleneck. Which of the following commands should you enter to stop the offending print job and free up the queue?
- lpq -s {job number}
 - lprm {job number}
 - lpq -s {print file name}
 - lprm {print file name}
9. You recently installed a new graphics card in one of your Linux systems that will be used by Develetech's video editor. The video editor notices sluggish performance and repeated crashes in their video editing software. You've deduced that the software isn't the source of the problem; the graphics card is. What step(s) should you take to try to solve this problem?
10. An administrator has reported that a Linux server is sluggish, unresponsive, and frequently triggers a kernel panic with the error message "Machine Check Exception". What can you do to diagnose and fix this issue?

- 11. One of Develetech's software developers has been programming on a Raspberry Pi device. He's been trying to connect the Pi's serial cable to a Linux system so that he can see the output of the program on the Linux console. However, the output doesn't appear. What can you do to troubleshoot this issue?**
-

Summary

In this lesson, you managed various types of hardware devices that can work with the Linux operating system. You configured these devices to fit your needs, and monitored them to ensure they are recognized and fully accessible by the system. You also identified any potential issues with hardware and developed solutions for those issues.

What type of storage and peripheral bus interface(s) do you use in your organization, or do you expect to use?

What are the most common hardware issues that you or your colleagues experience in the workplace?



Practice Question: Additional practice questions are available on the course website.

Lesson 10

Managing Networking

LESSON TIME: 4 HOURS

LESSON INTRODUCTION

Another major component of the operating system is networking. Networking is crucial for almost all modern systems, including those that run Linux®. In this lesson, you'll review some of the fundamental concepts that are involved in networking, and then you'll configure networking in your Linux systems. This will ensure your systems, especially servers, will be able to communicate with other computers both locally and around the world.

LESSON OBJECTIVES

In this lesson, you will:

- Identify the fundamental concepts of the TCP/IP networking protocol.
- Identify the roles that various Linux servers can play.
- Connect to a network.
- Configure DNS and DHCP client services.
- Configure cloud and virtualization technologies.
- Troubleshoot networking and connection issues.

Topic A

Identify TCP/IP Fundamentals

Before you start developing your Linux-specific networking skills, you need to review some of the fundamental concepts that comprise a modern TCP/IP network. This is essential knowledge that will prepare you for the configuration ahead.

TCP/IP

The networking process is governed by protocols. Much like human conversation, computer network communications are managed by rules. These rules control how much information is communicated at any given time, what addresses are used to represent nodes, how nodes take turns communicating on the network, what error checking may exist, etc. Many families or suites of protocols have existed over the years, but the only protocol suite of note today is **Transmission Control Protocol/ Internet Protocol (TCP/IP)**. This is the default protocol of the Internet and most internal private networks.

THE OSI MODEL

The **Open Systems Interconnection (OSI) model** standardizes how networking is meant to function. The OSI model contains seven layers. Each layer represents an area of responsibility that must be satisfied for the networking process to work. IT professionals need to be aware of how these layers work. You should memorize them, in order, with a brief definition of each layer.

The OSI Model also serves as a standard point of reference when communicating with other network professionals. For example, you may be instructed to purchase a "Layer 2 switch" or to troubleshoot a "Layer 3" problem. It serves as a common reference point for devices, and a guide for developers creating network applications.

Layer	Name	Basic Function
7	Application	Supports applications and end-users.
6	Presentation	Formats data for use.
5	Session	Establishes, maintains, and tears down a connection.
4	Transport	Enables reliable transmission of information.
3	Network	Enables logical addressing (IP addresses).
2	Data link	Enables physical addressing (MAC addresses).
1	Physical	Enables physical network connectivity.

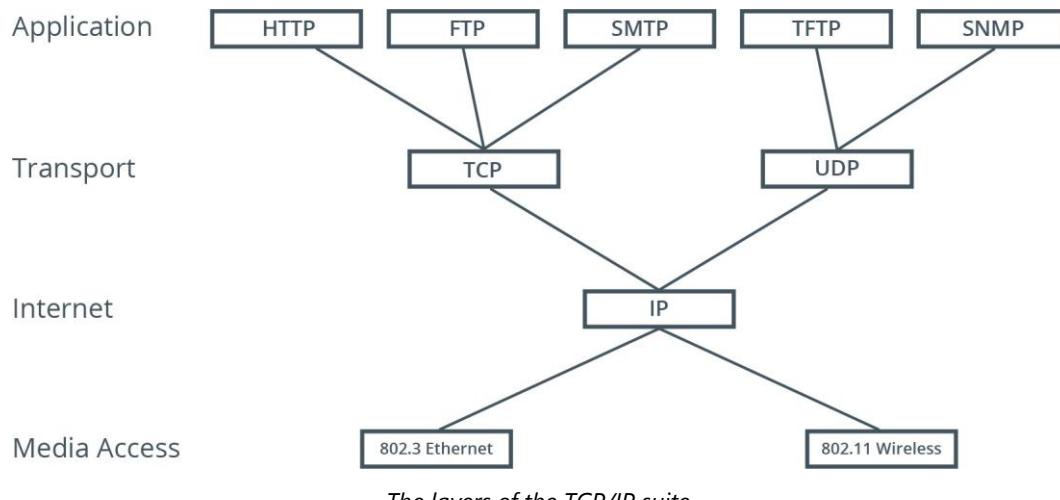


Note: There are a great many additional functions and details about the OSI model. As a networking professional, you must familiarize yourself with these functions. This table only serves as a brief overview.

TCP/IP LAYERS

The TCP/IP protocol suite satisfies the requirements of the OSI model in four layers. The suite is used to govern network communications on most internal networks. It is the protocol suite of the Internet, as well. TCP/IP is used by Linux, Unix®, macOS®, Windows®, etc. It is imperative that you have a basic understanding of the TCP/IP suite and its layers.

An understanding of the TCP/IP suite can aid in troubleshooting and network configuration. For example, IP address support occurs in the IP layer, while application support occurs at the application layer.



NETWORK IDENTITIES

The term "node" refers to devices with an identity on the network. That identity may be represented by a physical address or one of two logical addresses, or any combination of the three.

Identifier	Description
MAC address	Each network interface card (NIC) has a unique identity coded into it that identifies that NIC on network segment. That code is referred to as a media access control (MAC) address. It is the most fundamental network identity and is considered a physical address.
IP address	Each NIC may be assigned a logical address called an IP address. The IP address uniquely identifies the NIC in the network environment. IP addresses are shown in dotted decimal (base 10), which is a range of 0–9. The computer works with the IP address in binary (base 2), which is a range of 0–1.
Hostname	Nodes may be given a human-readable name that helps people better understand what device they are working with. This identity is often configured during the installation of the operating system and is sometimes called the "computer name." Hostnames are limited to 255 characters.



Note: Different devices on the network reference the different addresses. Often switches will govern traffic based on MAC addresses, while routers will manage traffic based on IP addresses.

NETWORK DEVICES AND COMPONENTS

There are several essential network devices and components to understand. These may be part of the troubleshooting process or network installation. Linux systems need to be configured properly to interact with these network devices and components.

Network Device/	Description
Switch	This device acts as a concentrator, centralizing all network connections for a segment to a single device. Switches can be used to manage traffic for performance and security concerns. As a general rule, switches work with MAC addresses at Layer 2 of the OSI model. There are switches that work at higher layers, too.
Router	This device acts as a control point for communications between network segments. Administrators can configure the router to permit or deny certain kinds of traffic, as well as pass traffic from one network segment to another. Routers work with IP addresses at Layer 3 of the OSI model.
Media	Typically, network cable is twisted pair Ethernet cable. Twisted pair may come shielded (STP) or unshielded (UTP). It is inexpensive and relatively easy to work with. It is the most common type of network cable. Other cable types include coaxial (coax) and fiber optic. Wireless networks forego cables and can transmit data over the air.

FRAME VS. PACKET

There are two main units of data that TCP/IP networks use in communications as the data moves up and down through the TCP/IP stack. When the data transmission is at the network layer (layer 3), it is referred to as a "packet." When the data is at the data link layer (layer 2), it is referred to as a "frame."

DNS AND DHCP

There are two network services that are commonly involved with TCP/IP network configuration. You will need to understand the role of these services in order to properly configure your Linux system.

Humans have a difficult time working with long strings of numbers such as IP addresses. The Domain Name System (DNS) service provides name resolution, a way of relating an easy-to-remember hostname with a difficult-to-remember IP address. DNS is implemented as a database hosted on one or more servers. The database may only contain the names and IPs of nodes in your own network, or it may be part of the larger Internet DNS infrastructure.

All nodes on the network must be configured with a unique IP address and other corresponding information. There are two ways of accomplishing this configuration—statically or dynamically. Static configuration is usually appropriate for servers and network devices, while dynamic configuration is typically used with end-user workstations. The Dynamic Host Configuration Protocol (DHCP) service provides dynamic configuration.

IPV4 ADDRESSING

IP addresses provide an addressing system for managing network identities. Internet Protocol version 4 was defined in 1981. The addresses are 32 bits in length, providing approximately 4.3 billion addresses. Humans usually work with IP addresses in the decimal form, such as 192.168.2.200, while network devices work with the address in binary.

IPv4 addresses are divided into at least two portions—a network identifier and a host identifier. The network identifier defines to which network segment the host belongs, and the host identifier uniquely identifies that host within the segment. Because the network ID may use different bits within the address, a second numeric value is used to show which portion of the IP address is the network ID and which part is the host ID. This value is known as the subnet mask. It is essential to understand the role of the subnet mask. It indicates where in the IP address the division is between the network ID and the host ID.

IPV4 CLASSES

The approximately 4.3 billion IPv4 addresses are divided into five classes. These classes provide a framework for possible segmentation of networks. Each class provides a specified number of networks, as well as a number of hosts available on each network. For the first three classes, the division between the network ID and the host ID occurs at one of the dots. Network professionals must be able to recognize all five classes by the value of the first octet, and know the default subnet mask for each class.

The 4.3 billion IPv4 addresses are divided into the following five classes.

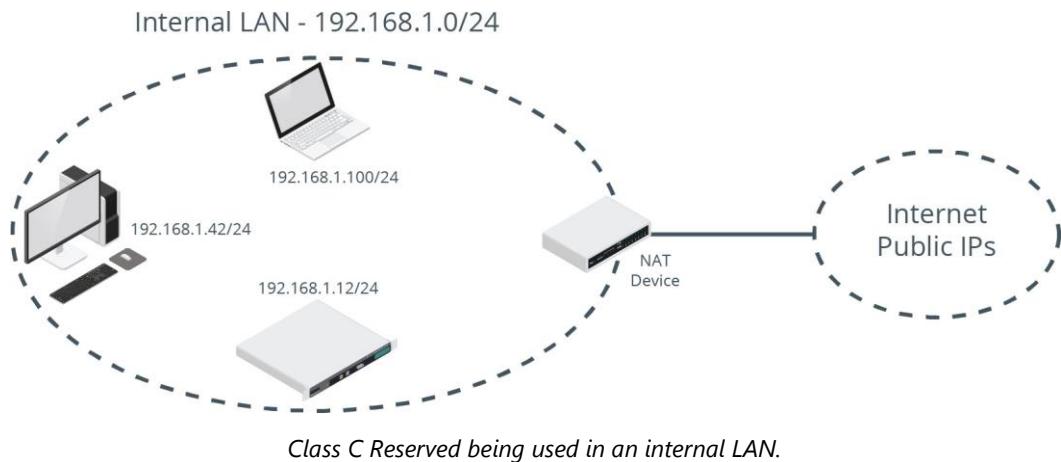
Class	Starting Address	Ending Address	Networks	Hosts Per Network	Default Subnet Mask
Class A	0.0.0.0	127.0.0.0	126	16,777,214	255.0.0.0 or /8
Class B	128.0.0.0	191.255.0.0	16,384	65,534	255.255.0.0 or /16
Class C	192.0.0.0	223.255.255.0	2,097,152	254	255.255.255.0 or /24
Class D	224.0.0.0	239.255.255.255	N/A	N/A	N/A
Class E	240.0.0.0	255.255.255.255	N/A	N/A	N/A

Note: Class D is multicast and Class E is experimental.

RESERVED RANGES

In addition to knowing the five IP address classes, there are several other IP addresses or address ranges that are important. Due to the depletion of IPv4 addresses, there are three IP address ranges that are reserved for internal use only. You will almost always find these in use on internal business and home networks.

- Class A Reserved: 10.0.0.0–10.255.255.255
- Class B Reserved: 172.16.0.0–172.31.255.255
- Class C Reserved: 192.168.0.0–192.168.255.255



LOOPBACK AND LINK-LOCAL

There are also two other IP address configurations to recognize. The loopback address is used for diagnostics purposes and to give the system the ability to network to itself. The link-local range is used for zero-configuration LANs or when the DHCP lease generation process fails. Link-local is also referred to as Automatic Private IP Addressing (APIPA).

- Loopback: 127.0.0.1
- Link-local: 169.254.0.0–169.254.255.255

IPV6

The IPv4 addressing scheme has many limitations. A newer standard is being implemented in the form of Internet Protocol version 6. IPv6 addresses many of the weaknesses of IPv4.

The main advantages of IPv6 over IPv4 are:

- IPv6 has a much larger address space.
- IPv6 has built-in encryption.
- IPv6 has more efficient routing.

Linux is fully compatible with IPv6, so Linux server and workstations should not be a limiting factor in the deployment of IPv6 in a network environment.

NETWORK PORTS

Network port numbers are numeric values assigned to the various application-layer protocols. Network devices use these port numbers to understand what application will handle the communication. Humans work with the application-layer protocols by name, such as Hypertext Transfer Protocol (HTTP). Computers need to work with these by port number.

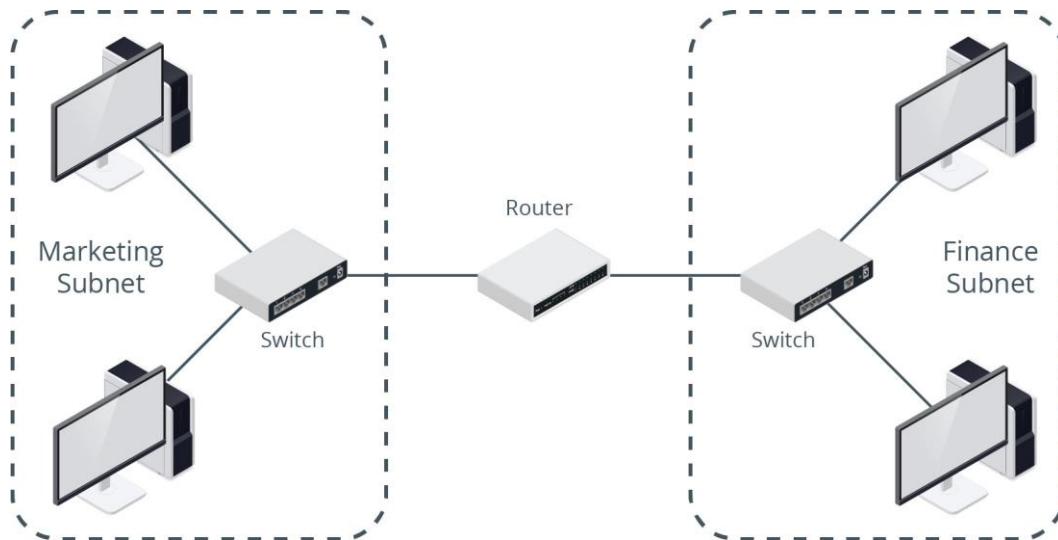
A few common port numbers are:

- 22: Secure Shell (SSH)
- 25: Simple Mail Transfer Protocol (SMTP)
- 80: Hypertext Transfer Protocol (HTTP)
- 110: Post Office Protocol version 3 (POP3)
- 443: Hypertext Transfer Protocol Secure (HTTPS)

NETWORK SEGMENTS

Network administrators will divide a network into segments in order to better manage network traffic. Their goal may be to manage that traffic more efficiently, resulting in better network performance, and/or to isolate that traffic for the purpose of security.

The logical divisions of the network are referred to as subnets and are identified by a network ID. This network ID is part of the IP address each node is using. All nodes in that subnet will have the same network ID in their IP address. Each node will have a unique host ID within that subnet. Recall that the subnet mask shows which part is the network ID and which part is the host ID.



Activity 10-1

Identifying TCP/IP Fundamentals

SCENARIO

Before you start configuring the networking on your Linux servers, you decide to review what you know about TCP/IP fundamentals. By discussing these key ideas, you'll be better equipped to manage networking on Linux.

- 1. List at least five application-layer protocols.**

- 2. What are the two transport-layer protocols?**

- 3. What protocol within the TCP/IP stack is responsible for logical addressing?**

- 4. What are the names and numbers of each layer of the OSI model? What is each layer's function?**

- 5. What is the name of the physical address used in Ethernet networking?**

- 6. What is the name of the logical address used by the TCP/IP protocol suite?**

- 7. What is the human-readable name of a network device?**

- 8. What are the two components of an IP address?**

- 9. What is the bit length of IPv4 addresses?**

- 10. List the address ranges for Class A, Class B, and Class C IP addresses.**

 - 11. List the default subnet mask for Class A, Class B, and Class C IP addresses:**

 - 12. What is the IP address associated with the loopback address?**

 - 13. What IP address range is associated with link-local/APIPA addresses?**

 - 14. What are two reasons why a network administrator might segment a network?**
-

Topic B

Identify Linux Server Roles



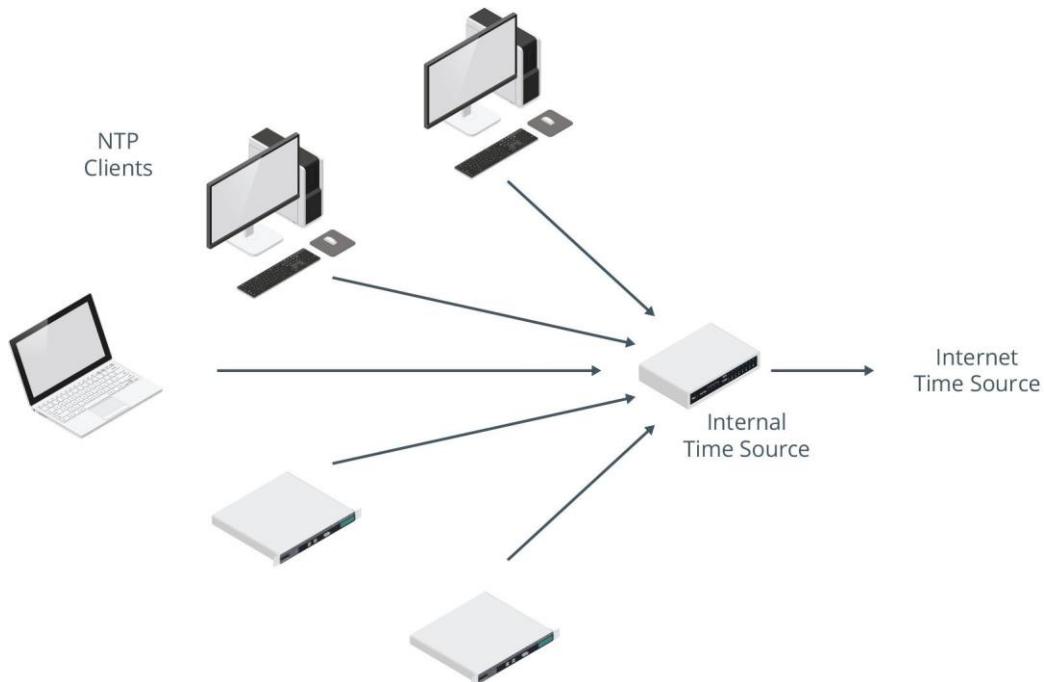
EXAM OBJECTIVES COVERED

2.5 Summarize and explain server roles.

In TCP/IP networking, specific protocols or categories of services are made available to clients on a network. Linux servers can be deployed to implement these services. So, in this topic, you'll identify some of the most common server roles.

NTP SERVICES

The **Network Time Protocol (NTP)** service enables the synchronization of a node's time with a designated, definitive time source. Time synchronization is essential in networking, making NTP configurations very important. Linux systems may be configured as NTP sources or NTP clients. NTP uses UDP port 123.

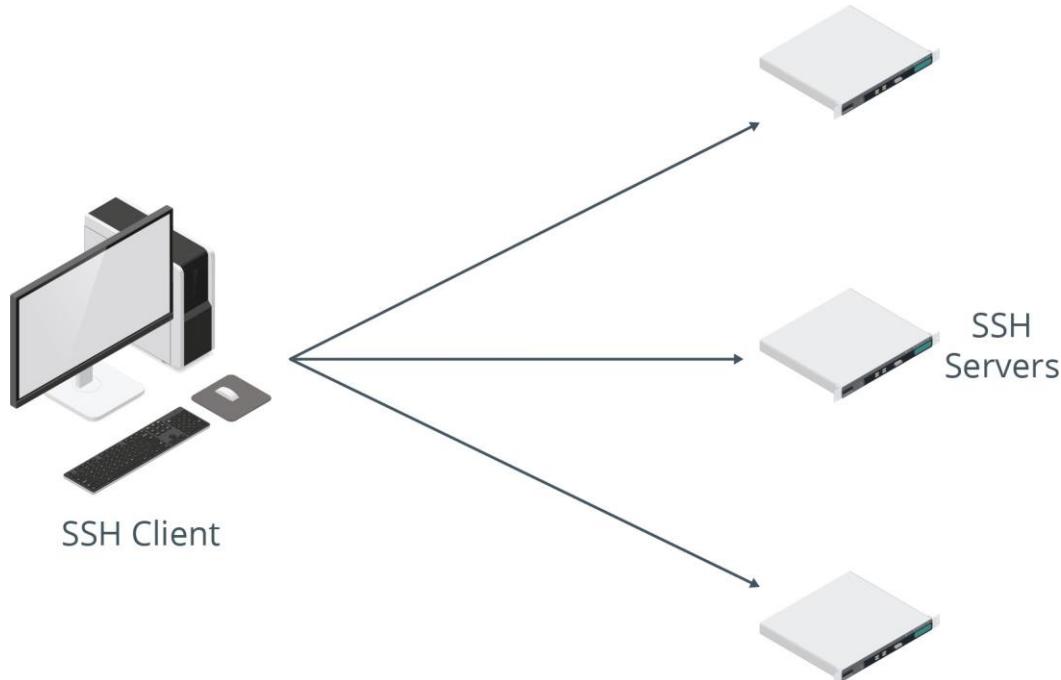


NTP clients synchronizing with an internal NTP server, which itself synchronizes with an NTP server on the Internet.

SSH SERVICES

The Secure Shell (SSH) service provides an authenticated, encrypted method of connecting to a remote (or even a local) system. Most frequently, SSH is used for remote administration, though it can be used as a tunnel to carry other kinds of network communications securely. SSH administration is very common in the Linux

and Unix worlds. Linux may be configured as an SSH client and/or server. SSH uses TCP port 22.



An SSH client connecting to remote SSH servers.

PUTTY

Microsoft Windows operating systems do not commonly use SSH, though it can be added to them. One of the most common Windows SSH clients is called PuTTY, and it is an excellent tool for those working with both Linux and Windows systems.

WEB SERVICES

Web servers host the files and images that make up websites. Client machines connect to the web server and download the files and images. Linux is a very common platform for hosting websites. Web services on Linux are typically hosted through a service called Apache.

Two protocols are primarily used with web services: **Hypertext Transfer Protocol (HTTP)** uses TCP port 80 and **Hypertext Transfer Protocol Secure (HTTPS)** uses TCP port 443.

CERTIFICATE AUTHORITY SERVICES

Certificates provide a way of guaranteeing identity. They are based on the use of a public key infrastructure (PKI) and asymmetric encryption. **Certificate authority (CA)** servers manage the enrollment, approval, expiration, and revocation of certificates.

One use of certificates is in guaranteeing the identity of websites for the use of HTTPS connections. Linux servers can be configured as certificate authorities.

NAME SERVER/DNS SERVICES

Name resolution is the relating of easy-to-remember hostnames with difficult-to-remember IP addresses. These relationships are typically stored in databases on **Domain Name System (DNS)** servers. A DNS server may contain records for a company's internal network, or it may be part of the Internet's name resolution

infrastructure. Linux systems may be configured as DNS servers and/or as DNS clients. DNS uses port 53 for both TCP and UDP.

DHCP SERVICES

Linux servers and workstations need to be properly configured to participate on the network. These configurations include an IP address, a subnet mask, default gateway (router), and other values. The configurations can be set manually, referred to as "static," or automatically, referred to as "dynamic." When a client is configured to retrieve an IP address dynamically, it does so by leasing the configuration from a **Dynamic Host Configuration Protocol (DHCP)** server.

Linux systems may be configured as DHCP servers, providing IP address configurations to other systems, or as a DHCP client, leasing an IP address configuration from a DHCP server. Typically, systems playing the role of server will have a static IP configuration, whereas client computers will have a dynamic IP configuration. The DHCP service uses UDP port 67 and 68.

SNMP SERVICES

Some network devices are capable of passing information about their performance and workloads to a central management database. These devices use the **Simple Network Management Protocol (SNMP)** service to accomplish this goal. Linux can act as a central management server for SNMP devices, or as an SNMP device itself. SNMP is not as common as it once was due to security concerns. SNMP uses UDP port 161 and port 162 for both TCP and UDP.

AUTHENTICATION SERVICES

Centralized authentication of user identities, rather than local authentication, makes the network more secure, simpler to manage, and easier for users to navigate. Authentication servers hold information about user identities in a directory store. When a user attempts to authenticate to a system, their login name and password are compared with what the authentication server has stored, instead of what the local workstation has stored. There are many kinds of authentication used by Linux. One example is Kerberos and another is Lightweight Directory Access Protocol (LDAP).

PROXY SERVICES

A proxy service resides on a system that has a direct connection to the Internet (an untrusted connection) and also an internal network connection (a trusted connection). The purpose of the proxy is to pass Internet requests between the two networks. One example of proxy services is web browsing. A client computer will pass a web request to a proxy, then the proxy will connect to the Internet to satisfy the request. The returned information comes to the proxy, which then passes the web data to the client machine that originally requested it.

Linux systems can be configured as proxy servers. Linux is often a good choice for this role because it can be run in a very lightweight configuration and is considered to be relatively secure. One common example of a proxy service for Linux is Squid. Squid has existed for a very long time and is frequently included with many Linux distributions.

LOGGING SERVICES

It is essential for Linux system administrators to be aware of occurrences on the servers they are responsible for. Log files can provide an immense amount of information about how the server boots, what services are running and how they are performing, what users may be doing on the system, etc. The traditional log file

mechanism for Linux has been syslog, though there are now several different logging services available.

One of the key functions of these logging services is to centralize log files from many Linux servers to one. This makes them easier to archive for service-level agreements (SLAs), troubleshooting and diagnostics, and performance auditing. With centralization, the Linux systems will forward their logs to a single server, which can then store all the log files. This long-term storage may be in the form of an SQL database or other database technology.

MONITORING SERVICES

There are many monitoring services available in Linux. Some monitor specific applications, like the Apache web service, while others monitor the Linux operating system itself. Whether or not these particular tools are installed by default will depend on which distribution you're using.

Examples of monitoring services include:

- **top**—monitors CPU and memory usage.
- **ApacheTop**—provides log file analysis for Apache, as well as information on connection response times, etc.
- **Monit**—a simple monitoring utility for Linux that watches hardware usage as well as directory and file information.
- **System Monitor**—the GNOME GUI tool for gathering information on system resource usage.

LOAD BALANCING SERVICES

Load balancing services are used to distribute inbound connection requests across multiple servers. A very common use for load balancing is to distribute connection attempts among web servers. A web server would be a single point of failure and could easily be overwhelmed by large amounts of traffic. Using multiple web servers alleviates these concerns, but a load balancing service is needed to ensure connections are spread across the available servers.

CLUSTERING SERVICES

On an internal network, access to services such as databases is essential to productivity. Data may reside on a storage area network (SAN). Access to the SAN and its stored content may be provided through a **cluster** of servers. Each server in the cluster is referred to as a node and can accept client connections. If one node in the cluster goes down, whether planned or unplanned, the other nodes can maintain availability.

FILE/PRINT SERVICES

File and print services are two of the most common network services; nearly every network has its foundation in basic file storage and print capabilities.

File servers, like those that use the File Transfer Protocol (FTP), enable the centralization of user data. Such centralization provides many advantages in a business network. These advantages include easier and more efficient backups, more secure storage of information, greater fault tolerance, and easier access to information. It is much easier to manage data on a single file server than to manage information that may be distributed across many end-user workstations.

Centralized printing services also include greater efficiency and significant cost savings. Print costs are measured on a price per page basis. A single large, very fast, very efficient network print device will be much less expensive on a per page basis than

individual print devices provided to each end-user. It is also much easier to manage paper and toner supplies for a few powerful print devices than for a great many individual print devices.

SAMBA AND NFS

Recall that Linux has a Server Message Block (SMB)-compatible file sharing protocol called Samba, as well as a separate file sharing protocol called NFS. Samba enables the integration of Linux and Windows systems through a protocol that is native to Windows. When added to a Linux workstation, that workstation can use the native Windows file and print sharing protocol to access shared resources on a Windows Server. When the Samba service is added to a Linux server, the server uses the native Windows protocol to share directories to Windows clients.

NFS, on the other hand, is a native Unix/Linux protocol used to provide workstations access to directories stored on a server. The centralization of files on a single server is highly desirable, because it makes physical security and backups much more straightforward. The primary configuration file for NFS is the `/etc/exports` file. This file is used to specify what directories are exported or made available on the network, as well providing access controls to those directories.

DATABASE SERVICES

Databases are used to store large quantities of data and to make it easier to query the database to retrieve the needed information. There are two database types frequently encountered in Linux environments—SQL and NoSQL database structures. **Structured Query Language (SQL)** databases use relational tables to relate information, whereas **NoSQL** databases do not organize information with relational tables. Examples of SQL databases include: MySQL™, MariaDB®, and PostgreSQL. An example of a NoSQL database is MongoDB®.

VPN SERVICES

Virtual private network (VPN) servers enable remote users to connect to the internal company network and access internal resources as if they were physically present at the network location. This is very useful for users who work from home or work from the road. Linux is capable of acting as both a VPN server and a VPN client.

VPN services are especially important because home and hotel networks are untrusted, as is the Internet. Content is encrypted within the VPN client computer before it is sent across the untrusted networks, then decrypted in the VPN server at the other end of the connection. Any data intercepted on the untrusted networks remains secure because of this encryption.

VIRTUALIZATION/CONTAINER HOST SERVICES

Virtualization has become essential to business services in the past decade. Virtualization is split into two general types: virtual machines and containers.

Virtual machines (VMs) rely on virtualization of the computer hardware. A hypervisor layer of software resides over the physical hardware and manages the allocation of that physical hardware to the virtual machines that are created. Operating systems, including Linux, can then be installed into the virtual machine. A virtual machine can provide the same full functionality of a traditional, physical server. VMs typically provide cost savings by more efficient use of the hardware along with many additional fault tolerance and management advantages. Examples of virtualization include Oracle® VM VirtualBox, VMware Workstation™, Microsoft Hyper-V®, and Kernel-Based Virtual Machine (KVM).

Containers operate with a different structure. A single host operating system runs multiple applications in isolation from each other, but all applications share the OS and its resources. Containers also provide a great deal of efficiency and management advantages. One example of a container service is Docker.

EMAIL SERVICES

Email services are key components of business communications. Email servers are responsible for the distribution of electronic mail within an organization or between organizations. Examples of email services for Linux include Sendmail and Postfix.

The common email protocols are:

- Simple Mail Transfer Protocol (SMTP) (TCP port 25)
- Post Office Protocol (POP3) (TCP port 110)
- Internet Message Access Protocol (IMAP) (TCP port 143)

Activity 10-2

Identifying Linux Server Roles

SCENARIO

The Develetech help desk has asked you to address some questions they've been receiving about network settings and configuration options. By answering these questions, you'll get a better idea of what server roles you may need to configure in order to meet Develetech's business needs.

1. **You want to provide secure connectivity to users in your company to a shared server for web and shell-based command-line access. Which protocols will you choose?**

2. **Your Windows users complain that there is no native SSH client in their older version of Windows. What can you recommend?**

3. **You need to provide shared Linux resources to Windows users. Which service can you set up on your Linux system to accomplish this?**

4. **Email is an essential network service. Which programs might you select for setting up email services on your Linux server?**

5. **Mobile devices have created a mobile workforce. Your users need to be able to connect to the corporate network to access shares, printers, and other internal tools. Which service can you set up to enable secure connectivity for your mobile users?**

6. **Rose Stanley, a graphic designer at Develetech, has asked for your help. She regularly transfers many very large image files. She wants an efficient way of doing this over the network. What network service do you recommend?**

7. **Chris Mason, a sales executive at Develetech, has been told to connect to a particular server to access resources. He has asked you for help because the instructions indicate he should connect by IP address. He cannot easily remember the IP address and has asked you if there is an easier way to connect to the server. What service can you use to make the connection easier for Chris?**

 8. **Develetech employees have contacted the help desk because they cannot connect to a particular server. Part of your investigation indicates the server has been configured with a dynamic IP address. How could this contribute to the problem and what steps should be taken to address it?**

 9. **Develetech's security team requires that log files be archived for future reference and audits. Since you are responsible for several Linux servers, how can you make this process more efficient?**

 10. **One of the developers at Develetech has asked for your help. She needs a Linux test environment to verify her application functions as designed. She would like to manage it herself and be able to revert it back to its original configuration after each test. What solution can you suggest?**
-

Topic C

Connect to a Network



EXAM OBJECTIVES COVERED

- 1.3 Given a scenario, configure and verify network connection parameters.
- 4.1 Given a scenario, analyze system properties and remediate accordingly.

Now that you're familiar with foundational networking concepts, you can start to configure networking on your Linux servers. You'll use various utilities to ensure that your systems are able to connect to the network and talk to other computers.

HOSTNAME CONFIGURATION

The systemd startup mechanism uses a command named `hostnamectl set-hostname` to configure the hostname value for the system. As with other services, once the configuration change is made, you must also restart the service. Here is an example of setting a new hostname with the `hostnamectl` command:

```
sudo hostnamectl set-hostname server01
```

IP CONFIGURATION

For a computer to participate on a network, it must have a valid identity as well as know the location of a few key services. The identities include a MAC address, an IP address, and a hostname. IP addresses must be configured on the system. Hostnames are normally configured during the installation of the operating system.

The IP configurations required include an IP address, the related subnet mask, the location of the default gateway (router), and typically the location of one or more name servers (DNS servers).

It is essential to verify this information early on in the configuration and troubleshooting processes. Any errors or misconfigurations in these values will usually result in the system not being able to participate on the network.

NetworkManager

Linux distributions often include a utility called **NetworkManager** to aid in the proper configuration of the IP information. NetworkManager includes three different interfaces that may be used, depending on whether or not a GUI is available on the Linux system.

THE nmcli COMMAND

The `nmcli` tool is the most fundamental of the NetworkManager interfaces. It contains many subcommands that enable you to view and configure network information. Because many network servers will not include a GUI, it is important to be comfortable with `nmcli` to manage network settings.

DEVICE	TYPE	STATE	CONNECTION
enp3s0	ethernet	connected	enp3s0
virbr0	bridge	connected	virbr0
wlp2s0	wifi	disconnected	--
lo	loopback	unmanaged	--
virbr0-nic	tun	unmanaged	--

Displaying the status of all connected network interfaces.

The following are some example of subcommands you can use with `nmcli`.

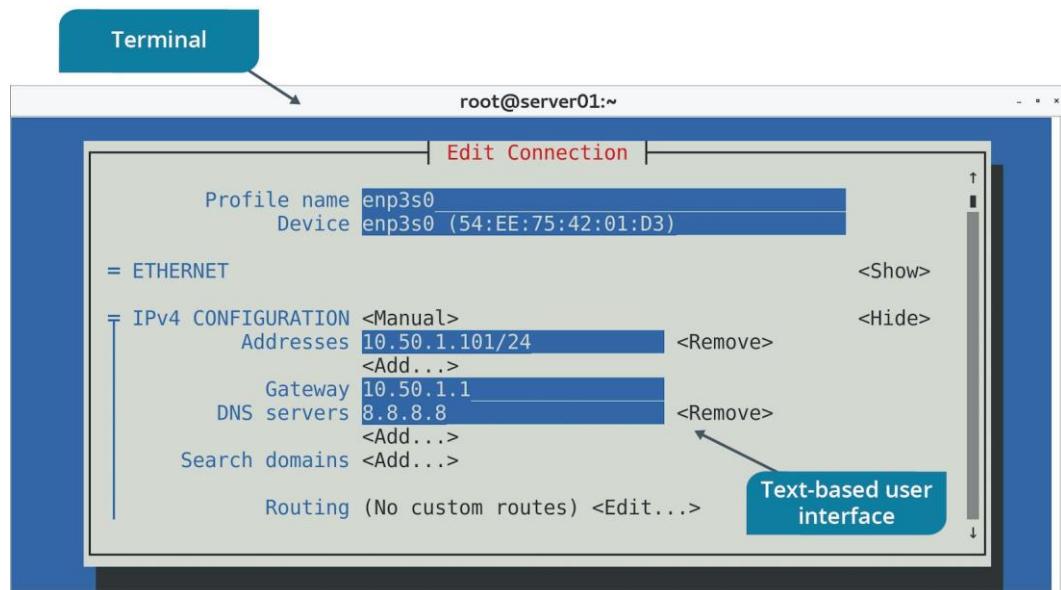
Subcommand	Used To
general status	View a summary of network connectivity data.
connection show	View identification information for each NIC.
con up {device ID}	Enable the specified NIC.
con down {device ID}	Disable the specified NIC.
con edit {device ID}	Enter interactive mode to configure the specified NIC.
device status	Display the current status of each NIC.

SYNTAX

The syntax of the `nmcli` command is `nmcli [options] [subcommand] [arguments]`

THE nmtui UTILITY

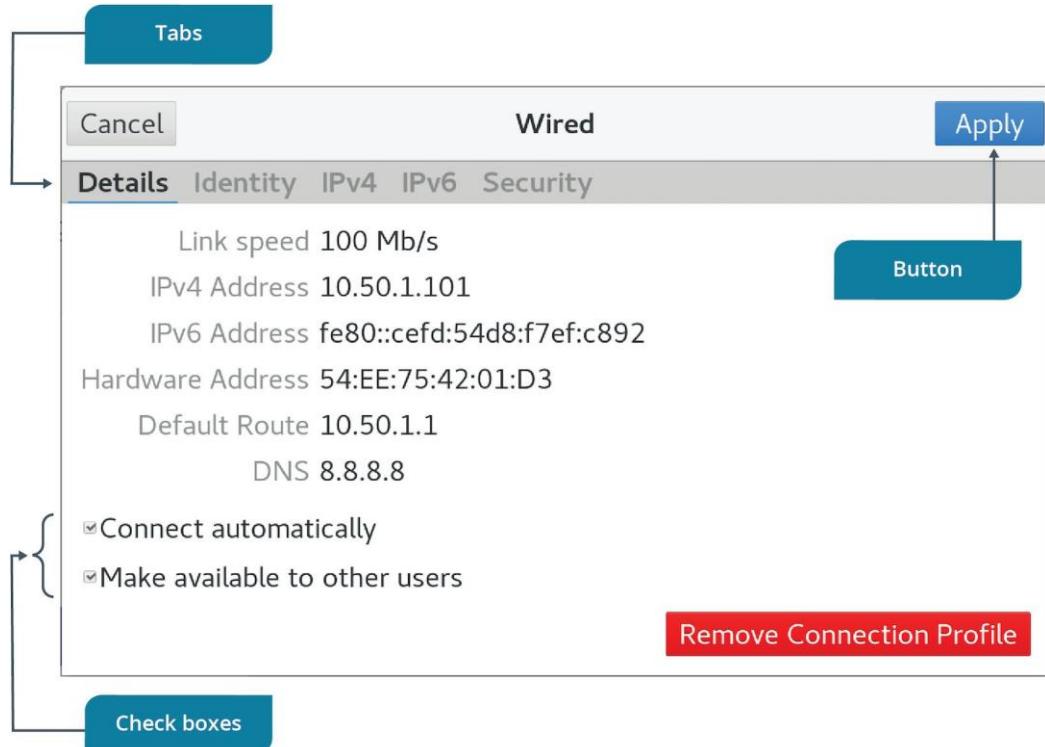
While Linux administrators often work at the command-line, it is certainly useful to have a visual representation of network configuration options. By running the `nmtui` command, you can call up a text-based user interface, or TUI. Navigating a TUI is accomplished by using the **Tab** key, the **Spacebar**, the **Enter** key, and the arrow keys. The **Tab** key moves the cursor from field to field. The arrow keys are used to make selections within the field. The **Enter** key is used to activate a setting, such as **OK** or **Quit**. The **Spacebar** is used to check or uncheck a check box.



Editing a network interface using a TUI.

THE nmgui UTILITY

NetworkManager also includes a GUI tool, which is particularly helpful for managing the network connections of workstations. The **nmgui** tool enables IPv4 and IPv6 configuration, as well as providing access to a wide variety of other network settings. This tool will certainly be familiar to most end-users.



Viewing network interface details in a GUI.

THE ifconfig COMMAND

The **ifconfig** command enables a user to view the current IP addressing information for each NIC recognized by the system. Viewing the IP address configuration is one of the earliest steps in network troubleshooting. The **ifconfig** command shows the IP address, subnet mask, broadcast ID, MAC address, basic performance information, and NIC name. The tool also enables NICs to be placed in an up or a down configuration (enabled or disabled).

The **ifconfig** command is officially deprecated in Linux, as noted in the man page; however, it is still available in many current distributions.

SYNTAX

The syntax of the **ifconfig** command is **ifconfig [options] [interface]**

THE ip COMMAND

The **ip** command replaces **ifconfig** in many distributions. It provides similar information to **ifconfig**, including IP address, subnet mask, MAC address, etc. The **ip** command will be one of the first tools used in network troubleshooting on a Linux system.

The following are examples of using the **ip** command:

- **ip addr show** —shows the IP address information on all interfaces.
- **ip link** —shows the status of each interface.
- **ip link set eth1 up** —enables the interface identified as **eth1**
- **ip link set eth1 down** —disables the interface identified as **eth1**

```

root@server01 ~]# ip addr show enp3s0
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 54:ee:75:42:01:d3 brd ff:ff:ff:ff:ff:ff
    → inet 10.50.1.101/24 brd 10.50.1.255 scope global noprefixroute enp3s0
        valid_lft forever preferred_lft forever
    → inet6 fe80::cefd:54d8:f7ef:c892/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@server01 ~]#

```

Data link details

Interface

IPv4 details

IPv6 details

Showing details for a network interface.

SYNTA X

The syntax of the **ip** command is **ip [options] {object} [subcommand]**

THE iwconfig COMMAND

The **iwconfig** command is used to provide wireless NIC configurations, including settings like SSID, encryption information, etc.

SYNTAX

The syntax of the **iwconfig** command is **iwconfig [options] [interface]**

iwconfig COMMAND OPTIONS

The following table describes some common **iwconfig** command options:

Option	Used To
<code>nick {name}</code>	Set a nickname for the NIC.
<code>mode {mode}</code>	Set the operating mode for the NIC that corresponds to the network topology.
<code>freq {number}</code>	Set the Wi-Fi frequency used by the NIC
<code>channel {number}</code>	Set the Wi-Fi channel used by the NIC.
<code>retry {number}</code>	Set the maximum number of MAC retransmissions for the NIC.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 10-3

Configuring the Server's Network Identity

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

You need to ensure the system's hostname and IP address configuration is correct. You also need to be able to configure network settings whether or not a GUI is installed. You will configure the system with both a static IP address and a dynamic IP address.

1. Set the server's hostname.
 - a) In a terminal window, enter **hostname** to view the system's current hostname.
 - b) Enter **nmcli general hostname** to use a different command to view the system's current hostname.
 - c) Enter **sudo hostnamectl set-hostname server##** to configure a new hostname.
Replace **##** with your student number. For example: **server01**
 - d) Enter **sudo systemctl restart systemd-hostnamed** to restart the service, making the change persistent.
 - e) Verify that your system's hostname has changed.

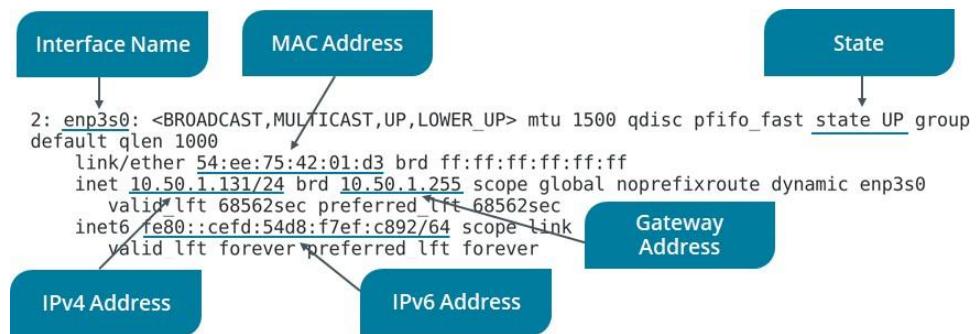
2. Verify the current IP address configuration of the server.
 - a) Enter **man ifconfig** and note the man page entry that indicates the tool is deprecated (retired).
 - b) Press **q** to quit.
 - c) Type **ip** (don't press **Enter**), add a space, then press **Tab** twice to see a list of available subcommands.

Note: Be sure to include a space before pressing **Tab** twice.



- d) Enter **ip addr** to display information about available network interfaces.
On CentOS 7, the main Ethernet device you should use will usually be named in the format **enp##s#**
For the following steps, make sure you're using this, and not the loopback adapter or a wireless LAN adapter.

- e) Enter **ip addr show <device ID>** to display the information for a specific NIC.



Use the NIC **<device ID>** from the output of the previous step. For example, the device ID value might be **enp0s3**



Note: One of the first steps in networking troubleshooting is to verify the current IP address configuration. Therefore, the **ip** command will be essential to your network troubleshooting process.

3. Display network information by using **nmcli**
 - a) Enter **nmcli general status** to view the current network connectivity status according to NetworkManager.
 - b) Enter **nmcli connection show** to see the name, UUID, type, and device ID for each NIC.
 4. Disable and enable a NIC using **nmcli**
 - a) Enter **nmcli con down <device ID>** to stop the NIC, making it inactive.
 - b) Enter **nmcli device status** to view the current status.
 - c) Enter **nmcli con up <device ID>** to re-enable the NIC, making it active.
 - d) Enter **nmcli device status** to view the current status.
 5. Configure the system with a static IP address using **nmcli**
 - a) Enter **ip addr show <device ID>** to view the current IP address.
 - b) Enter **nmcli con edit <device ID>** to edit the NIC's configuration.
 - c) Enter **set ipv4.addresses 10.50.1.1##/24** to set the static IP address at the **nmcli** prompt.
Replace **##** with your student number. For example: **10.50.1.101**
-
- Note:** Your instructor may provide you with different IP addressing information.
- d) Press **Enter** to set **ipv4.method** to **manual**.
 - e) Enter **save** at the **nmcli** prompt.
 - f) Enter **quit** at the **nmcli** prompt.
 - g) Enter **nmcli con down <device ID>**
 - h) Enter **nmcli con up <device ID>** to reset the device.

- i) Enter **ip addr show <device ID>** to confirm the static IP address is configured.

```
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 54:ee:75:42:01:d3 brd ff:ff:ff:ff:ff:ff
    inet 10.50.1.101/24 brd 10.50.1.255 scope global noprefixroute enp3s0
        valid lft forever preferred lft forever
    inet6 fe80::cefcd:54d8:f7ef:c892/64 scope link noprefixroute
        valid lft forever preferred lft forever
```

6. Configure the system as a DHCP client.

- a) Enter **nmtui** at the prompt to open a new interface.



Note: Use the **Tab** key and the arrow keys to navigate text-based user interfaces. Use the **Spacebar** to check/uncheck settings. Use the **Enter** key to accept a configuration.

- b) Make sure **Edit a connection** is highlighted, and then press **Enter**.
- c) With your device ID highlighted from the **Ethernet** menu, press the **Right Arrow** key once then the **Down Arrow** key to highlight <**Edit...**> and then press **Enter**.
- d) Notice the static IP address, as configured in the previous task.
- e) Press the **Tab** key three times to move to the **IPv4 CONFIGURATION** line.
That line currently displays <**Manual**>.
- f) Press **Enter** and select **Automatic** from the menu.
- g) Press the **Tab** key multiple times until you reach the bottom of the interface and <**OK**> is highlighted.
- h) Press **Enter** to save your changes to the network configuration.
- i) Use the **Tab** key to highlight <**Back**> and then press **Enter**.
- j) In the **NetworkManager TUI** interface, use the **Down Arrow** key to highlight **Quit** and then press **Enter**.
- k) Enter **ip addr show <device ID>** and notice that the old statically assigned IP address is still in place.
This is because you need to restart the network service for changes to take effect.
- l) Enter **sudo systemctl restart network**
- m) Enter **ip addr show <device ID>** and notice a new IP address is configured, leased from a DHCP server.

7. Using the GUI, reconfigure the NIC to use a static IP address.

- a) From the desktop menu, select **Applications**→**System Tools**→**Settings**.
- b) In the **Settings** menu, select **Network**.
Notice the NIC is displayed as **Connected** and **On**.
- c) Select the **Configuration** gear button.



Note: The NIC details may still show the static IP address.

- d) Select the **Apply** button in the upper-right corner of the interface.
- e) Select the slider to turn the NIC **Off**, then turn it back **On**.
- f) Select the **Configuration** gear button again and note the leased IP address.
- g) Select the **IPv4** tab.
- h) Observe that the **Automatic (DHCP)** button is selected, as configured in the previous **nmtui** task.
- i) Select the **Manual** radio button, and then fill in the **Address**, **Netmask**, and **Gateway** fields:
 - IP address: **10.50.1.1##**

- Subnet mask: **255.255.255.0** or **/24**
- Gateway: **10.50.1.1**



Note: Your instructor may provide you with different IP addressing information.

- j) In the **DNS** field, enter **8.8.8.8**
This is one of Google's DNS servers.
- k) Select **Apply**.
- l) Select the slider to turn the NIC **Off**, then turn it back **On**.
- m) Close the **Settings** window.
- n) Test the network configuration by opening **Applications**→**Favorites**→**Firefox Web Browser** and browsing to the **https://www.comptia.org** website.
- o) When you're done, close the browser.

THE ethtool COMMAND

The **ethtool** is used to manage NIC driver and network configurations. Whether or not it is installed by default will depend on the distribution in use. The **ethtool** utility has a great many options for gathering information.

```
root@server01:~# ethtool -S enp3s0
NIC statistics:
tx_packets: 34038
rx_packets: 41021
tx_errors: 0
rx_errors: 0
rx_missed: 0
align_errors: 0
tx_single_collisions: 0
tx_multi_collisions: 0
unicast: 39280
broadcast: 1741
multicast: 0
tx_aborted: 0
```

Showing statistics for a network interface.

SYNTAX

The syntax of the **ethtool** command is **ethtool [options] {device name}**

ethtool COMMAND OPTIONS

The following table describes some common **ethtool** command options.

Option	Used To
-S {interface}	Show statistics for a NIC.
-i {interface}	Show driver information for a NIC.
-t {interface}	Execute a self-test on the NIC.
-s {interface} {setting} {value}	Change some of a NIC's settings, such as its speed and duplex mode.

Option	Used To
-f {interface} {image}	Write ("flash") a firmware image to the NIC.

THE brctl COMMAND

Network bridging involves associating two networks that normally would not pass network traffic between them. Bridging works at OSI Layer 2 with MAC addresses. A Linux system can be configured to bridge two networks. The **brctl** (bridge control) command is used to configure bridging within Linux.

A common example of bridging is as follows:

1. **brctl show** —View the bridging configuration.
2. **brctl addbr {bridge name}**—Create an empty bridge.
3. **brctl addif {bridge name} eth0** —Add **eth0** to the bridge.
4. **brctl addif {bridge name} eth1** —Add **eth1** to the bridge, linking the networks connected to **eth0** and **eth1**.

SYNTAX

The syntax of the **brctl** command is **brctl [command]**

NIC BONDING

Associating two or more NICs together on the same system enables aggregation of their combined bandwidth, fault tolerance/redundancy, and load balancing. The two interfaces would normally be managed independently and have different IP address configurations. When using **bonding**, the interfaces are managed as a single device and have the same IP address configuration. The ability to bond two or more NICs is an important feature for a server. If there are two NICs in a bond, and one fails, the other is present to continue providing connectivity.

NIC bonding can be configured in an active/passive setup. In this setup, one primary NIC is active, and a secondary NIC is on standby (passive). If the active NIC fails, the system automatically fails over to the secondary NIC to maintain availability. NIC bonding can also be configured in a load balancing manner, where the combined bandwidth of each is used in a way that one of the NICs is not overwhelmed. The NICs that are members of the bond are referred to as "slaves."

NIC bonding is supported in Linux, but support must also be present in the network switch. This support is called EtherChannel. It is a feature of most enterprise-class switches.



Note: Some Linux distributions also support NIC teaming, which is a similar but more flexible approach to link aggregation. For example, RHEL 6 supports NIC bonding, whereas RHEL 7 supports both NIC bonding and NIC teaming.



Two NICs bound on the same switch.

NIC BONDING PROCESS

The process for creating a NIC bonding configuration begins with creating the bond, then adding NICs to the bond.

1. Ensure the kernel module that supports NIC bonding is loaded.
2. Run the `modprobe --first-time` binding command to load the module.
3. Run the `modinfo` binding command to see additional information on the kernel module.
4. Create a file named `bond.conf` in the `/etc/modprobe.d/` directory with the following content: `alias bond0 bonding`
5. Create the bond interface:
 - a. `nmcli con add type bond con-name bond00 ifname bond00 mod active-passive`
6. Add the NIC interfaces to the bond interface:
 - a. `nmcli con add type bond-slave con-name bond00- <device ID> ifname <device ID> masterbond00`
 - b. `nmcli con add type bond-slave con-name bond00- <device ID> ifname <device ID> masterbond00`
7. Set the `bond00` interface with its IP configuration as if it were a regular NIC. For a server, this usually means you will set a static IP address.
8. Test the connection, including any failover settings. Don't forget that you may also need to configure the switch to support NIC bonding.



Note: NIC bonding can be configured using `nmcli`, `nmtui`, or the NetworkManager GUI tools.

THE /etc/sysconfig/network-scripts/ DIRECTORY

The `/etc/sysconfig/network-scripts/` directory contains network device configuration files. These files include the configurations of any NICs, bonds, and bridges that might exist on the Linux system. These files usually take the form of `ifcfg-<NIC>`. Settings can include whether the NIC is configured for static or dynamic IP addresses, whether the NIC is enabled or not, etc. The exact settings vary depending on the needed configuration and device type.

While it is possible to manually edit these files with a text editor like Vim or nano, the NetworkManager utility is often a much better way of managing the interfaces. There is a command-line, text interface, and graphical interface for NetworkManager.

```
root@server01 ~# ls -l /etc/sysconfig/network-scripts/
total 248
-rw-r--r--. 1 root root 351 Dec 14 21:24 ifcfg-enp3s0 ← Interface configuration file
-rw-r--r--. 1 root root 254 Jan  2 2018 ifcfg-lo
lrwxrwxrwx. 1 root root 24 Dec 11 18:17 ifdown -> ../../sbin/ifdown
-rwxr-xr-x. 1 root root 654 Jan  2 2018 ifdown-bnep
-rwxr-xr-x. 1 root root 6569 Jan  2 2018 ifdown-eth
-rwxr-xr-x. 1 root root 6190 Apr 11 2018 ifdown-ib
-rwxr-xr-x. 1 root root 781 Jan  2 2018 ifdown-ippp
-rwxr-xr-x. 1 root root 4540 Jan  2 2018 ifdown-ipv6
lrwxrwxrwx. 1 root root 11 Dec 11 18:17 ifdown-isdn -> ifdown-ippp
-rwxr-xr-x. 1 root root 2102 Jan  2 2018 ifdown-post
-rwxr-xr-x. 1 root root 1068 Jan  2 2018 ifdown-ppp
-rwxr-xr-x. 1 root root 870 Jan  2 2018 ifdown-routes
-rwxr-xr-x. 1 root root 1456 Jan  2 2018 ifdown-sit
-rwxr-xr-x. 1 root root 1621 Mar 17 2017 ifdown-Team
-rwxr-xr-x. 1 root root 1556 Mar 17 2017 ifdown-TeamPort
-rwxr-xr-x. 1 root root 1462 Jan  2 2018 ifdown-tunnel
lrwxrwxrwx. 1 root root 22 Dec 11 18:17 ifup -> ../../sbin/ifup
-rwxr-xr-x. 1 root root 12415 Jan  2 2018 ifup-aliases
```

The `/etc/sysconfig/network-scripts/` directory.

THE `/etc/network/` DIRECTORY

For Debian-derived distributions, network configuration files representing the interfaces can be found in the `/etc/network/` directory. Many Debian-based distributions also use NetworkManager, so editing the files in `/etc/network/` is usually not necessary.

NETPLAN

Netplan is a network configuration utility found on some distributions. It uses YAML description files to configure network interfaces. These files are stored in the `/etc/netplan/` directory. You will use a text editor to create or modify interface configuration files. Netplan includes subcommands to make configurations more reliable. For example, you can enter `sudo netplan` to have the configuration file checked for syntax errors before attempting to implement it. The `sudo netplan apply` command actually makes the configuration change.

THE `/etc/sysconfig/network` FILE

The `/etc/sysconfig/network` file is used to configure whether networking should be enabled at boot, as well as hostname information, gateway information, etc. These settings may instead be configured on a per-interface basis in the `/etc/sysconfig/network-scripts/ifcfg-<NIC>` files.



Note: The `/etc/sysconfig/network` file typically exists in Linux distributions that are derived from Red Hat Linux. For Linux distributions derived from Debian, the equivalent file is `/etc/sysconfig/interfaces`



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 10-4

Discussing Connecting to a Network

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **You are setting up a new Linux server on your corporate network. For a computer to participate on a network, what must the server possess?**

2. **Your manager has set up a new project for you and your team to give descriptive hostnames to all Linux servers. How can you set a system's hostname to fswebserver01?**

3. **You ask a coworker to check the IP address on server03. He attempts to use ifconfig but it isn't installed. How can you instruct him to check the IP address?**

4. **One of the administrators on your team prefers working with graphical user interfaces (GUIs) rather than at the command-line. However, there are no GUI elements installed on your Linux servers. He needs to configure some additional networking elements for several systems. Which command can you advise him to use?**

5. **On Linux systems that have a GUI installed, you configure network settings using a NetworkManager graphical utility. What is the name of that utility?**

Activity 10-5

Verifying Network Configurations

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. Previously, you configured networking information on your server.

SCENARIO

Now that you've configured a NIC, you need to verify that those configurations are active and accurate. So, you'll use **ethtool** and the device's configuration file to confirm the networking details.

1. Gather information with **ethtool**
 - a) If necessary, enter **ip a** to recall your Ethernet device ID.
 - b) Enter **ethtool <device ID>**
 - c) Verify that you can see information about the NIC's capabilities and configurations.
You should be able to see the NIC's maximum bandwidth speed, its duplex capabilities, its supported link modes, and more.
2. View network configuration files.
 - a) Enter **ls /etc/sysconfig/network-scripts** to display the contents.
 - b) Verify that there is a **ifcfg-<device ID>** file.
 - c) Enter **cat /etc/sysconfig/network-scripts/ifcfg-<device ID>** to view the contents.
 - d) Verify that you can see device information as well as IP addressing information for this NIC.

Topic D

Configure DHCP and DNS Client Services



EXAM OBJECTIVES COVERED

1.3 Given a scenario, configure and verify network connection parameters.

As you've seen, DHCP and DNS are both significant protocols in the world of TCP/IP networking. In this topic, you'll take a deeper dive into how these protocols are implemented in Linux and how you can configure them to suit your needs.

STATIC VS. DYNAMIC IP ADDRESS CONFIGURATION

IP address configurations can be set two ways: statically or dynamically. Each method has advantages and disadvantages. It is important to know when to use each method.

Static IP address configuration means that the settings are implemented manually by an administrator. This method increases the risk of mistakes, but also ensures the system always has the same identity. Static IP address configurations are usually appropriate for network devices (routers, network switches, etc.) and servers. They may also be used for network print devices.

Considerations for static configurations:

- IP address configuration never changes.
- Possibility of mistakes when entered manually.
- Administrative workload is increased for configuration and reconfiguration.

Dynamic IP address configuration means that the settings are retrieved from a server. This method decreases the risk of mistakes, but also means the system may not always have the same IP address. Dynamic IP address configurations are usually appropriate for client machines. They may also be used for network print devices.

Considerations for dynamic configurations:

- IP address configuration may change over time.
- Reduces the risk of typographical errors during configuration.
- Administrative workload is decreased for configuration and reconfiguration.

DHCP CONFIGURATION

A Dynamic Host Configuration Protocol (DHCP) server contains one or more pools of IP addresses and related configuration options that client machines can lease. This saves a significant amount of administrative effort, since IP address information does not have to be configured statically for each system.

DHCP servers are configured with a scope, or a range of available IP addresses, along with additional options. The DHCP service must be installed on the server that will host the service and allow client machines to lease configurations.

THE DHCP LEASE GENERATION AND RENEWAL PROCESS

Workstations lease their IP address configurations from a DHCP server. The process consists of four steps, initiated by the clients. The leases are temporary, so periodically

the clients will be required to renew their leases. The renewal process provides an opportunity to update the clients if there have been network configuration changes.

The lease typically includes the IP address for the client, the associated subnet mask, the IP address of the default gateway (router), the IP address of one or more DNS servers, and the length of time the lease is valid. There are additional options that may be configured as well.

Periodically, DHCP clients must renew their leased IP address configuration. If the DHCP server detects that the client has out-of-date information, it will force the client to lease a new configuration. If there have been no changes, the renewal will succeed. The lease renewal process is steps three and four of the initial lease generation process.



The process of a DHCP client leasing an IP address from a DHCP server.

THE /etc/dhcp/dhclient.conf FILE

The primary DHCP client reference file is the `/etc/dhcp/dhclient.conf` file. This file enables the configuration of DHCP client settings, including timeout values, dynamic DNS configurations, etc. The file is called and managed by NetworkManager, which serves as a network configuration service for multiple network settings, including the DHCP client configuration. It is typically appropriate to manage DHCP client configurations by using NetworkManager rather than editing the `/etc/dhcp/ dhclient.conf` file directly.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

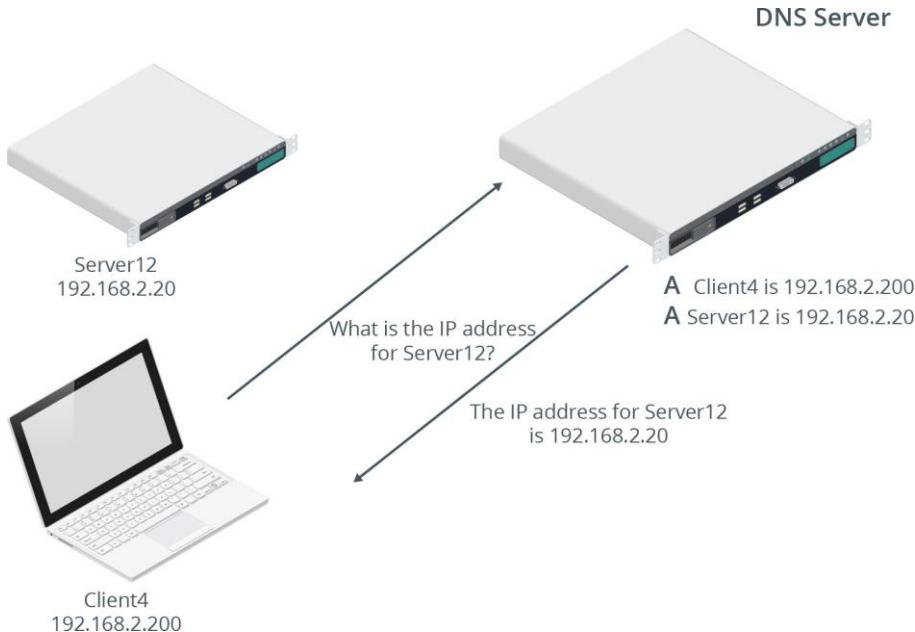
NAME RESOLUTION

TCP/IP data packets must include a source IP address and a destination IP address.

Humans have a very difficult time remembering long strings of numbers. For example, imagine if every website you have bookmarked or every email address you have in your contacts information was noted for you as an IP address instead of a name!

Humans work with easy to remember names, such as **www.redhat.com** or **www.ubuntu.com**. Such information is virtually useless to computers, which need the IP address information in order to properly find the resource. Names are a description, whereas addresses are a location.

Name resolution is the process of relating these easy-to-remember names with difficult-to-remember IP addresses. There are two general ways in which name resolution works. The first is via static text files such as the **/etc/hosts** file. The second method is via a dynamic database called Domain Name System (DNS).



The name resolution process using a DNS server.

TESTING NAME RESOLUTION

Various tools can be used to test name resolution, including the **host** and **nslookup** commands.

THE /etc/hosts FILE

Early network environments were relatively small and the identities of network nodes did not change frequently. Name resolution could be managed by using a text file, stored locally on each system, that contained all the systems and their related IP addresses. This file would have to be manually updated if there were any name or IP address changes, additions, or deletions. In today's modern, transient networks, this method is not realistic. The number of entries in the file and the frequent changes to the identity entries would be overwhelming.

The **/etc/hosts** file is still important, however, because it can be used in special case situations where a particular system—perhaps a developer's workstation—needs to connect to an experimental server that is not registered on the network. While the **/etc/hosts** file is not commonly used, it is essential in certain scenarios.

```
root@server01 ~]# cat /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.50.1.102 server02
```

IP address Hostname to resolve

Configuring name resolution locally.

THE /etc/resolv.conf FILE

Modern networks use a name resolution service like DNS to relate computer names and IP addresses. Network devices will query a DNS server in order to resolve a name and IP address relationship. The DNS server contains resource records that will provide answers to the query. DNS servers are centralized and much easier to maintain and keep current than **/etc/hosts** files are.

The **/etc/resolv.conf** file is stored locally on each system, informing that system of the IP address of one or more DNS servers. DNS is an essential service, so it is a good practice to have at least two DNS servers listed in the **/etc/ resolv.conf** file.

```
root@server01 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
nameserver 8.8.8.8
nameserver 8.8.4.4
```

DNS servers

Configuring name resolution using public DNS servers.

THE /etc/nsswitch.conf FILE

The **/etc/nsswitch.conf** file includes several configuration options. The option related to name resolution defines the order in which name resolution methods will be used by the system. The order may be the **/etc/hosts** file first, then DNS; or DNS first, then the **/etc/hosts** file. The preferred configuration is **/etc/hosts** then DNS.

```
root@server01:~  
[root@server01 ~]# cat /etc/nsswitch.conf | grep hosts  
#hosts:      db files nisplus nis dns  
hosts:      files dns myhostname
```



The */etc/nsswitch.conf* file.

DNS CONFIGURATION USING NetworkManager

DNS configurations can also be set using NetworkManager. NetworkManager includes command-line, text-based, and graphical interface utilities. In all three utilities, it is essential to configure the IP address of at least one DNS server.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 10-6

Discussing DHCP and DNS Client Services

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You have set up two internal DNS servers in your network and need to check if they're working to resolve hostnames to IP addresses and IP addresses to hostnames. Which utility can you use to do this?

2. Your team set up a VPN connection between the main corporate office and one of your branch offices, but you cannot connect to any system in the branch office network. One of the servers at the branch office has an IP address of 192.168.1.10. Which command can you use to check the network path between your computer and that server to help troubleshoot the problem?

3. You installed the Apache web service on Linux server web01, but you can't connect to the web service (TCP port 80) from another computer. To begin troubleshooting, you use a utility to check TCP connections and ports that web01 is listening on. Which utility can you use?

4. One of the other administrators on your team patched and rebooted a server, but is worried because it is taking a long time for the server to become available again. Which utility can you use to check when the system comes back online?

5. You suspect that someone is attempting to hack into one of your servers, so you decide to capture a few minutes of network traffic on that system to investigate further. Which command-line utility can you use to capture the traffic?

Activity 10-7

Configuring a DNS Client

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You will work with a partner in this activity.

SCENARIO

In addition to setting up machine-friendly IP addressing, you also need to account for the fact that humans aren't good at remembering long strings of numbers. So, you'll configure name resolution to relate a hostname with an IP address so that users can easily refer to a specific computer on the network.

1. Review the IP address and hostname identities of your system.
 - a) Enter **hostname** to view the system's user-friendly name.
 - b) Enter **ip addr show <device ID>** to view the system's IP address.
Humans don't tend to be good at remembering long strings of numbers. Name resolution is used to relate the hostname and the IP address values displayed above.
 - c) If the leased IP address is still visible, use **nmcli con down <device ID>** and then **up** to reset the interface.
2. Configure name resolution for your system.
 - a) Enter **cat /etc/resolv.conf** to display the DNS server(s) the system is configured to query.
 - b) Enter **cat /etc/hosts** to display the static text file that can be used for name resolution.
 - c) Ask your partner for the hostname and IP address of their system. Provide the same information to them about your system.
 - d) Enter **ping <partner hostname>** and verify that it fails.
 - e) Enter **ping <partner IP address>** and verify that it succeeds.
 - f) Press **Ctrl+C** to interrupt the process.
 - g) Using **sudo**, open the text editor of your choice to add your partner's hostname and IP address information into the **/etc/hosts** file in the format **<partner IP address> <partner hostname>**
For example: **10.50.1.101 server01**
 - h) Save and close the file.
 - i) Ping your partner's hostname and IP address again and verify that, this time, both succeed.
3. Ensure name resolution for Internet identities is functioning correctly.
 - a) Enter **host www.google.com**
 - b) Enter **nslookup www.google.com**

- c) Verify that you receive IP addressing results for **google.com** with each command.

```
[student01@localhost ~]$ host www.google.com
www.google.com has address 172.217.12.132
www.google.com has IPv6 address 2607:f8b0:4006:801::2004
[student01@localhost ~]$ nslookup www.google.com
Server:      8.8.8.8
Address:     8.8.8.8#53
```

Non-authoritative answer:
Name: www.google.com
Address: 172.217.12.132

Topic E

Configure Cloud and Virtualization Technologies



EXAM OBJECTIVES COVERED

1.5 Compare and contrast cloud and virtualization concepts and technologies.

One modern day concept that heavily leverages networking is the idea of cloud computing. Cloud computing has revolutionized the IT industry, as has the closely associated concept of virtualization. In this topic, you'll configure these technologies to optimize your Linux infrastructure.

CLOUD COMPUTING

A relatively new and rapidly changing aspect of IT is the growth of **cloud computing**. There are many examples and uses of cloud computing. According to the National Institute of Standards and Technology (NIST), there are five essential characteristics of cloud computing:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

These five concepts combine together to form one of the most important aspects of today's IT industry. Cloud computing has become a mission-critical service for many businesses, large and small alike. At its most fundamental, you might think of cloud computing as a method of off-loading responsibility. That responsibility might be for the cost, ownership, and maintenance of the network and server infrastructure, or responsibility for the development environment, or responsibility for operating system and application support.

Cloud services are characterized by flexibility of deployment, scale, support, and fault tolerance. Businesses are able to rapidly expand their resources on demand, and only pay for the resources that are actually consumed. Subscription-based payment provides greater financial flexibility and control. In a traditional infrastructure, where network and server components are purchased by the organization, resources may go unused for long periods of time or not be able to grow quickly enough to support business agility.

CLOUD MODELS

There are three primary models of cloud computing, with many variations of each.

Model	Description
Software as a Service (SaaS)	SaaS provides applications to end-users. The applications are not installed directly on the user workstation, although to the user it seems as though the application is local. Deployment, maintenance, upgrades, and security patches are off-loaded to the cloud service provider. This area of cloud computing typically provides service to all end-users.
Platform as a Service (PaaS)	PaaS includes virtualization of the operating system layer of the environment. The development or database platform is supported by the cloud service provider and given to the customer in a ready-to-use manner. Support for the platform is off-loaded to the cloud service provider. This area of cloud computing typically provides service to developers and database administrators.
Infrastructure as a Service (IaaS)	In IaaS, the physical devices (servers, switches, routers, etc.) are virtualized and owned by the cloud service provider. Responsibility for the hardware lifecycle is off-loaded to the cloud service provider. This area of cloud computing typically provides service to systems administrators.

 **Note:** There are several more models that target more specific services.

PUBLIC, PRIVATE, AND HYBRID CLOUDS

Cloud services can be provided with different structures. Public clouds refer to hardware resources that are owned by the cloud service provider, and customer resources, compute functions, storage, etc., may be shared among multiple customers. Laws, industry regulations, and company security policies may not permit all companies to use such shared resources. Private clouds refer to the use of cloud technologies as an on-premise solution. Private clouds often satisfy security requirements, enable a company to retain a great deal of control over exactly where their data resides at any given moment, and may provide easier management. Private clouds do require that the company continue to own the hardware lifecycle, application support, etc. Hybrid clouds are a combination of the other two concepts, enabling more effective cost management combined with strict security management.

 **Note:** Private clouds may also be known as local clouds, or on-premise cloud deployments.

CLOUD SERVICE PROVIDERS

Many cloud service providers (CSPs) exist, but Amazon and Microsoft cloud solutions are the two primary providers in the cloud industry. Amazon Web Services (AWS™) supports a great many deployment options. AWS is built on a Linux solution.

Microsoft® Azure® also supports a great many deployment options. Azure supports the deployment of Linux-based solutions, not just Windows solutions.

Red Hat® Cloud Suite is another CSP. Red Hat's solution is a Linux-based, full-featured private cloud.

Although AWS and Microsoft Azure are considered the industry leaders in cloud services, there are a great many additional CSPs. Many provide specialized services for

particular parts of the cloud market. Here are a few additional CSPs for you to investigate:

- **Google Cloud™**: Supports a very wide variety of services, including compute, storage, database, IoT, development tools, etc.
- **Rackspace**: Services include compute, database, business continuity, and data center management.
- **Heroku™**: Provides PaaS options for application development.
- **Digital Ocean™**: A PaaS cloud provider that provides scalability and management.

CLOUD AND VIRTUALIZATION

Virtualization enables significantly more efficient use of hardware, plus many additional options for fault tolerance, disaster recovery, and scalability. It also allows for much quicker deployment of servers and services. It is for these reasons that virtualization is the foundation of cloud computing. Cloud providers virtualize the resources they make available to customers to make management of those resources easier.

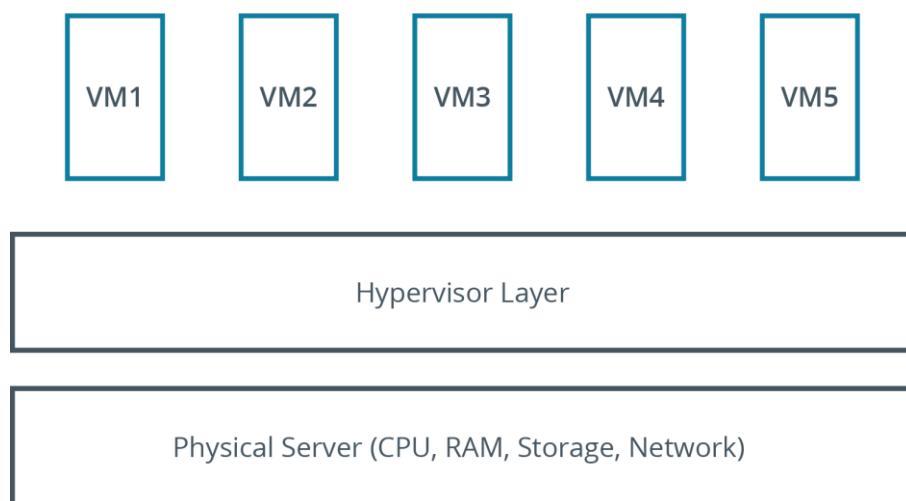
This virtualization extends to every model: SaaS, PaaS, and IaaS. For example, an administrator can easily build and later tear down an entire cluster of systems with a single action, or even automate this process in response to changes in the environment. It would be much less efficient if the administrator had to do all of this on physical systems, so virtualization is essential.

HYPERVISORS

The **hypervisor** software layer provides control between the virtual machines and the physical hardware. It manages allocation of hardware, control of networking, administrative control, etc.

There are two types of hypervisors: Type 1 runs directly on the hardware in a "bare metal" deployment. Examples include VMware ESXi and Microsoft Hyper-V. Type 2 runs as a service on a locally installed operating system. Examples include Oracle VM VirtualBox and QEMU.

Kernel-Based Virtual Machine (KVM) is a Linux-based virtualization solution that can be added to most Linux distributions. It enables the management of virtual machines on Linux platforms with attributes of both type 1 and type 2 virtualization.



The architecture of a type 1 hypervisor.

TEMPLATES

Virtual machine templates can make deployments much more efficient. Administrators and users can deploy servers themselves in a self-service environment using pre-defined templates that specify different processor, memory, storage, and network configurations.

There are different templates that can be used to make deployments easier.

Template	Description
Open Virtualization Format (OVF)	This contains the necessary configuration files, packages, etc., for virtual machines and network devices. These files may be used in the deployment of virtual machines in a virtualized environment. In addition, OVF files are easily distributed. The metadata of the virtual machine, including information about the VM's requirements and dependencies, is described in an XML-formatted file with a .ovf extension. Open Virtualization Appliance (OVA) refers specifically to a single package representing the network appliance, usually stored in a different format, such as a .tar file.
JavaScript Object Notation (JSON)	These files may be used to store information that is easy for most programming languages to interpret and use. Because JSON uses a standard JavaScript® format, it is also relatively easy for humans to interpret and write. One use of JSON files is for the quick deployment and configuration of one or more virtual machines.
YAML Ain't Markup Language (YAML)	This may be used to store configuration information that is used on newly deployed virtual machines. This information may be used by cloud-init to install software or create user accounts during the virtual machine's first boot, or may be used by orchestration tools like Ansible. YAML files consist of a list of key-value pairs that specify the desired configuration.
Container images	Many virtualization solutions use virtual machines (VMs). VMs are complete, standalone deployments atop the virtualization host. Containerization virtualizes at the operating system layer, providing application isolation, even though the applications are sharing a single OS. Containers may be useful for hosting production applications as well as for testing applications by developers. You can deploy containers using container images (e.g., Docker containers) that you have created yourself or have downloaded from the Internet. These images may contain everything needed for the container, including applications and supporting configurations.

BOOTSTRAPPING

The term bootstrapping refers to the startup of the operating system. It refers to the adage "pulling yourself up by the bootstraps," meaning that the operating system starts with simple layers that move toward more complex functionality. With virtual machines, the bootstrapping steps are handled by the virtualization layer. It is possible to modify the startup and the deployment of virtual machines during the bootstrapping sequences. Three ways of managing bootstrapping include cloud-init, Anaconda, and Kickstart.

Cloud-init is a cloud-based Linux mechanism to customize a virtual machine during its first bootup. This customization might include security settings, software package installations, user and group creation, etc. Cloud-init references YAML files to find the necessary settings. Cloud-init is supported by many distributions, including Ubuntu® Server 18 and Red Hat® Enterprise Linux® (RHEL) 7.

Many Linux distributions use the Anaconda installer to manage their deployments. This installer can provide prompts to an administrator to configure the new Linux server, or it can reference files to customize the installation. Linux uses Kickstart files to customize the installation, providing an unattended install. All information about partitions, packages, user accounts, software deployments, etc., are contained in the Kickstart file. The combination of Anaconda and Kickstart enables rapid, consistent, and customized Linux installations.

STORAGE

With traditional physical computers, storage capacity is provided as physical space. This space is usually one or more storage drives. With virtualization, storage capacity may be exposed to the virtual machines as virtual storage drives. The virtual storage drive is a file that resides on the physical drive, much like any other data file. That file is treated as if it were a physical drive. It is partitioned and given a file system, a boot loader is installed, and an operating system is deployed. While from the administrator's perspective the virtual drive is just a file, from the virtual machine's perspective, it's a physical device.

Because virtual drives are just files, they may provide more opportunity for fault tolerance, simpler rollback of a system's status, redundancy, and storage scalability as compared to a traditional physical drive.

THIN VS. THICK PROVISIONING

When deploying virtual machines, you may be offered the option of configuring thin storage or thick storage. Thin storage refers to a virtual storage device file that will grow on demand up to a maximum size. This may make for more efficient use of drive space, but it may also include a performance hit. Thick provisioning of a virtual storage device immediately reserves the allocated space for use by the virtual device only, regardless of whether that much capacity is actually needed. Performance is better, but it may consume more drive space than it needs. Thin provisioning is most appropriate in environments where the cost of maintaining large storage pools is much more of a concern than the risk of temporarily running out of storage. Likewise, thick provisioning is most appropriate in environments where disruptions to the continuous storage process present an intolerable risk, more so than the expense of unused storage.

PERSISTENT VOLUMES

Some organizations will manage container virtualization with Kubernetes. Kubernetes provides an orchestration solution for container management. As part of that orchestration, persistent volumes are created. When used in conjunction with a Linux cluster, they keep the storage configuration separate from the configurations of the

individual cluster nodes. This makes it easier for nodes to be replaced through their lifecycle without impacting the storage.

BLOB AND BLOCK STORAGE

Data may be stored in multiple ways. One way is to use traditional SQL databases, like MariaDB® or MySQL™. It is also possible to store data in an unstructured manner, which is referred to as a **blob**. Blob stands for binary large object. Audio, video, and other multimedia files—as well as text files—may be stored in this manner. This is an example of object storage.

Data itself is written to the storage device (whether physical or virtual) in small chunks called blocks. These blocks are the fundamental storage areas of the drive. Most files are too large to be stored in a single block, so the files are broken into pieces that are then written to the blocks. Data is reassembled when called by the operating system.

Object storage may be more efficient than block storage, especially for larger files. It may also be more flexible when storing data in multiple geographical regions.

NETWORKING CONFIGURATIONS

The virtualization hypervisor can be configured to provide access to networking services in several ways. The virtual machine can be configured with one or more virtual NICs. The virtual NICs can then be connected to virtual switches within the hypervisor. Those virtual switches may then be given access to the host computer's physical NICs.

Some network configuration options include:

- **No networking:** Simulates a computer that does not include a NIC at all.
- **Internal:** The VM is connected to a virtual switch that permits network communication with other virtual machines, but not network communication with the host operating system or the physical NIC.
- **Private:** The VM is connected to a virtual switch that permits network communication with other virtual machines and with the host operating system, but no network connectivity to the physical NIC.
- **Public:** The VM is connected to a virtual switch that permits network communication with other virtual machines, the host operating system, and the physical NIC. This configuration exposes the VM to the business network, where it can participate as if it were a regular client or server.

Network configuration for virtual machines must be carefully managed, because VMs frequently are configured as production servers to which end-users need access. Once network connectivity is provided, it is essential to follow best practices for securing network-connected nodes. The fact that the network node is a virtual machine does not prevent it from needing protection from network threats.

ADDITIONAL NETWORKING CONSIDERATIONS

Networking in virtualized environments is as flexible as it is in physical environments. Services such as Network Address Translation (NAT) provide the same functionality. In the case of NAT, that's to translate between the reserved, private IP address ranges and the public, Internet address ranges. These virtualized networks may be thought of as "overlay networks," especially when deployed in a cloud environment. The term overlay network simply means one network built over the top of another. Virtualized network environments also support bridging, which connects two networks into a single managed unit.

Virtualization supports the use of multiple NICs in virtual machines as well. If the NICs are connected to different network segments, the server is referred to as being "dual-homed."

VIRTUALIZATION TOOLS

Many virtualization host servers will run Linux without a graphical user interface. Such a configuration enables more hardware resources to be available to the virtual machines. Management of the virtual machines must then occur at the command-line.

The **virsh** command is an interactive shell to KVM virtual machines. The following are some subcommands you can use within **virsh**.

Subcommand	Used To
help	Get help with the virsh command.
list	Get a list of recognized VMs.
shutdown {VM}	Gracefully shut down a VM.
start {VM}	Start a VM.
reboot {VM}	Reboot a VM.
create {XML file name}	Create a VM from an XML file.
save {VM} {file name}	Save the state of a VM with the given file name.
console {VM}	Open a console to a VM.

libvirt

Linux virtualization solutions are built on top of libvirt, an application programming interface (API) that provides the software building blocks for developers to write their own virtualization solutions. Solutions can also be composed of a daemon and a management interface. Several hypervisors, including VMware ESXi, KVM, QEMU, etc., are all built using libvirt. It provides a solid foundation for Linux-based virtualization. For example, the **virsh** tool is a part of the libvirt API.

GNOME VMM

The GNOME Virtual Machine Manager (VMM) utility can be used for managing connectivity to virtual machines. It enables the deployment, management, and removal of virtual machines using an intuitive graphical interface. Download and install the **virt-manager** package to begin using VMM.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 10-8

Discussing Cloud and Virtualization Technologies

SCENARIO

Answer the following questions to check your understanding of the topic.

1. Your CIO has tasked you with the job of finding an appropriate cloud service to host some of your applications. The application needs to appear as if it is locally installed. Which cloud model will you select?

2. Executive management has decided to move toward a more agile infrastructure with mobile users and 100 percent hosted applications and infrastructure. This option enables employees to work from any location and relieves the company of hardware lifecycle and application support. Which option has the executive management decided to move to?

3. To move to a public cloud offering for your company's applications, you respond with the recommendation of using one of the two primary cloud service provider (CSP) companies. What are the two primary CSP companies from which to choose?

4. Your company needs to transition from individuals running virtual machines locally on workstations to deploying virtual machines company-wide. Your team determines that it is more cost-effective for you to use VMware ESXi. VMware ESXi is which type of hypervisor? What does this mean?

5. **There is much debate among your team concerning whether to deploy virtual machines with thick-provisioned storage or thin-provisioned storage. What the debate boils down to is performance. Which provisioning type has better performance?**
-

Activity 10-9

Configuring Virtualization

DATA FILE

```
/opt/linuxplus/managing_networking/yum-script.sh  
/opt/linuxplus/managing_networking/CentOS-7-x86_64-DVD-1804.iso  
/opt/linuxplus/managing_networking/ubuntu-vm.qcow2
```

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

One of the developers at Develetech has asked for your help. She needs Linux test environments to test that her application functions as designed. She'd like to manage the environments herself and be able to revert back to their original configuration for each test. You will install a KVM virtualization solution for her.

1. What are some of the potential benefits of virtualization?
 2. Install the KVM virtualization software.
 - a) Enter `cat /proc/cpuinfo | grep vmx` and then enter `cat /proc/cpuinfo | grep svm` to check the processor.
If either term is found, the processor should support virtualization.
 - b) Enter `Sudo yum -y install qemu-kvm qemu-img virt-manager libvirt libvirt-python libvirt-client virt-install virt-viewer bridge-utils librbd1 librbd1-devel libsolv`
This installs KVM and dependent software.
 - c) Wait for KVM to finish installing.
 3. Start the KVM service.
 - a) Enter `Sudo systemctl start libvирtd` to start the service.
 - b) Enter `Sudo systemctl enable libvирtd` to make the service persist.
 - c) Enter `lsmod | grep kvm` and verify that the KVM kernel module is loaded.
 4. Create a VM at the CLI.
 - a) Enter `Sudo virt-install --name=devtech-install --vcpus=1 --memory=2048 --cdrom=/opt/linuxplus/`

```
managing_networking/CentOS-7-x86_64-DVD-1804.iso
--disk size=12 --os-variant=rhel7
```

This defines the hardware specifications of the virtual machine to create. The VM will use one virtual CPU, have access to 2 GB of RAM, use the provided system image to boot from, and have access to a 12 GB storage drive.

- b) Close the VM window that pops up.
- c) Enter **`sudo virsh save devtech-install saved-vm`** to stop the VM and save its state for later.

- 5. Import a VM image using the GUI Virtual Machine Manager.
 - a) From the desktop menu, select **Applications**→**System Tools**→**Virtual Machine Manager**.
 - b) Enter the root password to continue.
 - c) In the Virtual Machine Manager, select **File**→**New Virtual Machine**.
 - d) In the **New VM** wizard, for the first step, select **Import existing disk image**, then select **Forward**.
 - e) For the second step, select **Browse** and then select **Browse Local**.
 - f) From the navigation menu, select **+ Other Locations**.
 - g) Select **Computer**.
 - h) Navigate to **/opt/linuxplus/managing_networking** and open **ubuntu-vm.qcow2**.
 - i) Select **Forward**.
 - j) For the third step, change the **Memory (RAM)** to **2048** and ensure **CPUs** is set at **1**.
 - k) Select **Forward**.
 - l) For the fourth step, name the VM **ubuntu-vm** and select **Finish**.

- 6. Get acquainted with Ubuntu, a different distribution of Linux.
 - a) Verify that a virtual machine window automatically pops up.
 - b) On the authentication screen, log in to the Ubuntu virtual machine using **student** as the account and **Pa22w0rd** as the password.
 - c) Verify that you successfully signed in to the Ubuntu desktop.



Note: If you receive an error that there is no space left on the device, reboot CentOS and try again.

- d) From the bottom-left corner, select the **Show Applications** button . You might need to scroll the VM window down to locate this button.



Note: If at any time you're prompted by the **Software Updater** dialog box, select **Remind Me Later**.

 - e) Select the **Settings** icon.
 - f) In the **Settings** window, from the navigation menu, select **Network**.
 - g) Select the configuration gear icon for the **Wired** connection to view the Ubuntu VM's networking information.
 - h) Close the **Wired** window, then close the **Settings** window.
 - i) From the **Show Applications** menu, select the **Utilities** icon.
 - j) Select the **Logs** icon.
 - k) Observe the log files that are displayed, then close the **Logs** window when you're done.
 - l) From the dock on the left side of the desktop, select the **Ubuntu Software** icon.
 - m) At the top of the window, select the **Installed** tab.
 - n) Scroll down and verify that Vim is installed, then close the window when you're done.

7. Shut down the virtual machine.

- a) Close the virtual machine window.
- b) Right-click the **ubuntu-vm** VM and select **Shut Down**→**Shut Down**.



Note: You may need to issue the shut down command twice.

- c) Wait for the VM's state to change to **Shutoff**.
 - d) Close Virtual Machine Manager.
-

Topic F

Troubleshoot Networking Issues



EXAM OBJECTIVES COVERED

- 1.3 Given a scenario, configure and verify network connection parameters.
- 4.1 Given a scenario, analyze system properties and remediate accordingly.

Networking issues are unavoidable in any organization, from small to large. In this topic, you'll use many different Linux tools that can diagnose and help you find solutions to problems like latency, disconnections, unresolved hostnames, and more.

COMMON NETWORKING ISSUES

Network troubleshooting begins with a series of general steps before moving to more complex techniques.

First, check the basics: Is the device powered on? Is the device plugged in? Are the cables properly connected to the correct network devices (switches, routers, etc.) and the hosts themselves? Also, remember that NICs can be in an "up" or a "down" state. Verify that they are available. Next, verify that the network interfaces are configured correctly. They need to have an appropriate IP address configuration, subnet mask, default gateway value, and name resolution settings. Also check that the network interface is detected by Linux and use the `ethtool` command to gather driver and other information about the device.

You can troubleshoot name resolution issues using several different techniques. The first is to ping a destination by both hostname and then by IP address. If the hostname-based ping fails, but the IP address-based ping succeeds, then the system is not properly resolving network names. At that point, you can use tools like `host` and `nslookup` to further test name resolution.

The network itself may also be experiencing issues. These issues might include very high traffic, causing latency and saturation issues at various points within the network. If a router is misconfigured or down, then the optimal path may not be found between two systems, or no path may be found at all. Network interface cards that are failing could cause data packets to be dropped on the network, causing a loss in data, timeout messages, or reduced network throughput. You may need to escalate these problems to the network team at your organization.

APPLICATION PERFORMANCE TROUBLESHOOTING

When troubleshooting application performance, developers may need to choose how communication with a service occurs. One method, `localhost`, creates a full network connection, including all TCP error checking, etc. This method may be significantly slower. The alternative design for developers is Unix sockets (or Unix domain sockets). This approach will often provide a performance increase by removing the TCP overhead from the transaction. The overhead isn't needed because the connection is entirely local.

NETWORK ADAPTER TROUBLESHOOTING

As you know, network adapters, also called network interface cards (NICs), provide physical connectivity to the network. These cards will typically have an RJ-45 connector. They may be found integrated on the motherboard or added to the board via an

expansion slot. Network cards require the appropriate driver to be installed so that Linux can take advantage of their features.

Some NICs support remote direct memory access (RDMA) interfaces, which are specialized hardware components implemented on very low-latency, high-speed networks to provide a significant performance increase. Support for RDMA is provided through a kernel module, so you will have to check the kernel on your server to see if the module is already loaded. In addition, you will need to provide a driver for the RDMA device if the kernel does not already contain one.

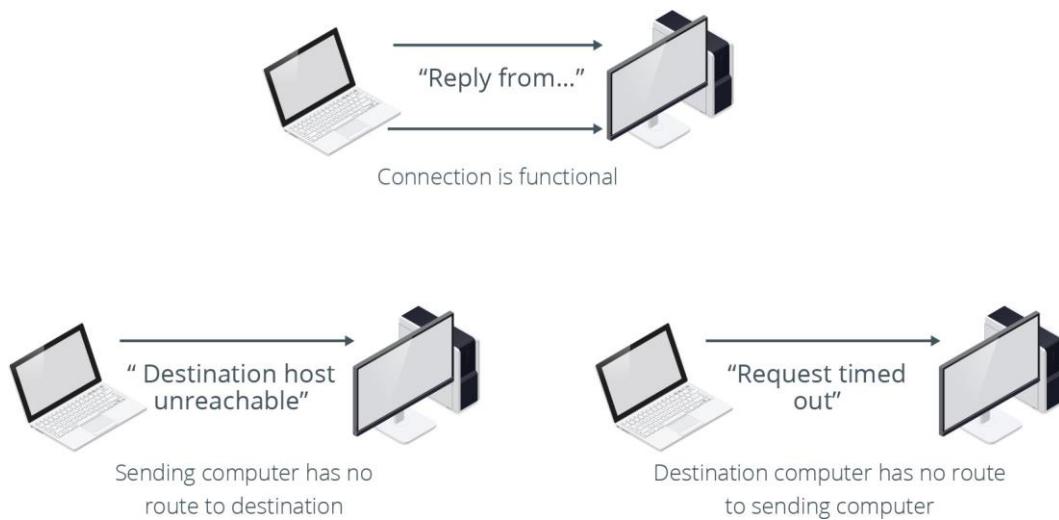
THE ping COMMAND

One of the earliest parts of network troubleshooting is sending test packets between two systems. This is done using a TCP/IP utility called **ping**. The **ping** command will generate a response request from the sending computer and should receive a reply from the destination computer.

Possible outcomes of the **ping** command include:

- **Reply from <host>**: The connection was successful.
- **Destination unreachable**: The source computer cannot find a path to the destination. This often indicates the problem is with the source computer.
- **Timeout**: The request reached the destination computer but a response did not return to the source computer before the source computer timed out. This often indicates the problem is with the destination computer.

Although using **ping** is one of the earliest steps in the network troubleshooting process, it only tells you that something is wrong—not *what* is wrong.



The possible results of pinging across a network.

SYNTAX

X

The syntax of the **ping** command is **ping [options] {destination}**

The **{destination}** can be an IP address, such as 192.168.1.1, or it can be a hostname, such as **server01**.

ping COMMAND OPTIONS

Some common **ping** command options include:

- **-C** —only send a specified number of pinging attempts. By default, Linux sends a continuous ping until interrupted with **Ctrl+C**.
- **-V** —specify verbose output.

SEQUENCE NUMBER

The ping command also provides a sequence number (`icmp_seq`) for each ping attempt. The host sending the ping can use this number to match each request with its response. Mismatched sequence numbers might indicate a dropped packet.

THE traceroute AND tracepath COMMANDS

The **traceroute** command is used to report the network path between the source and destination computers, including any routers the connection uses. The process of a packet traveling from one router to the next is called a hop. The **traceroute** command therefore outputs each hop along the path. This is particularly effective when troubleshooting Internet connectivity or connections within very large routed environments. If the **traceroute** fails, being able to identify where along the path it failed is useful for troubleshooting.

The **tracepath** command is a simplified version of **traceroute** that does not require administrative privileges to run. It also contains fewer options.

```
[root@server01 ~]# traceroute -I -m 10 comptia.org
traceroute to comptia.org (198.134.5.6), 10 hops max, 60 byte packets
 1  gateway (10.50.1.1)  0.511 ms  0.804 ms  1.697 ms
 2  192.168.38.1 (192.168.38.1)  3.895 ms  4.009 ms  4.114 ms
 3  rrcs-24-213-135-1.nys.biz.rr.com (24.213.135.1)  5.465 ms * *
 4  * * *
 5  * * *
 6  * be28.albynyyf01r.northeast.rr.com (24.58.32.70)  12.137 ms  12.311 ms
 7  bu-ether16.nycmny837aw-bcr00.tbone.rr.com (66.109.6.74)  18.020 ms  21.875 ms  22.
045 ms
 8  66.109.5.119 (66.109.5.119)  17.791 ms  58.801 ms  58.934 ms
 9  38.142.136.185 (38.142.136.185)  18.192 ms  18.292 ms  18.378 ms
10  qwest.iad01.atlas.cogentco.com (154.54.10.26)  17.758 ms  17.969 ms  17.845 ms
```

Following the route between two network hosts.

SYNTAX

The syntax of the **traceroute** and **tracepath** commands is **traceroute/tracepath [options] {destination}**

ROUTING ISSUES

Many routing issues are the result of misconfigured routing tables. These issues can usually be fixed by updating the routing tables. However, you must first identify what is causing the issue. Commands like **traceroute** and **tracepath** can reveal routing issues like routing loops, in which traffic is continuously routed back and forth between multiple nodes and never reaches its destination. For example, node A uses node B as a path to node C; but node B uses node A as a path to C. If traffic is bound

for node C, nodes A and B will endlessly send the traffic between them because they both think each other is the path to C.

THE netstat COMMAND

The **netstat** (network statistics) command is used to gather information about TCP connections to the system. Depending on the options used, **netstat** informs the user of existing connections, listening ports on the server, NIC information, etc.

Common options for the **netstat** command include:

Option	Used To
-v	Activate verbose mode.
-i [interface]	Display information about all network interfaces or the specified interface.
-c	Continuously print information every second.
-l	Show only what ports are being listened on.

The **netstat** command has been deprecated in favor of the **ss** command, but it may still be installed with some Linux distributions.

SYNTAX

The syntax of the **netstat** command is **netstat [options]**

OUTPUT

The default output of **netstat** is in columnar format, as follows:

- The protocol used by the socket.
- The number of processes attached to the socket.
- Flags that give further information about the socket's status.
- The type of socket access.
- The state of the socket.
- The ID of the process attached to the socket
- The path of the process attached to the socket.

An example is as follows:

```
unix 2 [ ] STREAM CONNECTED 472 /run/dbus/system_bus_socket
```

THE ss COMMAND

The **ss** (socket state) command is an information gathering utility similar to **netstat** but provides simpler output and syntax. The **ss** tool can provide information about established TCP connections or which ports the system may be listening on for inbound connections. This can help you diagnose problems related to clients and servers being unable to communicate with one another over the desired protocol; a missing socket could mean that the service isn't running, and a closed socket could mean that either the client or the server is prematurely terminating the connection.

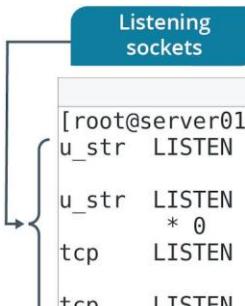
Another way to use **ss** is to gather information about a particular client that may be connected.

Common options for the **ss** command include the following.

Option	Used To
-l	Show currently listening sockets.

Option	Used To
dst {host}	Show whether the specified host is connected and what the connection statistics are.
-i	Show only what ports are being listened on.

Listening sockets



```
root@server01:~# ss -l | grep ssh
u_str  LISTEN      0      128  /tmp/ssh-anqcWhPB1Axb/agent.2329  35364
u_str  LISTEN      0      128  /run/user/1000/keyring/ssh  37090
tcp    LISTEN      0      128  *:ssh                           *:*
tcp    LISTEN      0      128  :::ssh                          :::*
```

Displaying listening sockets related to SSH.

SYNTAX

The syntax of the **ss** command is **ss [options]**

THE dig, nslookup, AND host COMMANDS

One of the most important network services is name resolution. If systems are not able to reach or use DNS name resolution servers, they are not likely to be able to access the needed network services.

Three different tools can be used to test and troubleshoot name resolution. These tools should be used after it has been confirmed that the system is configured with the proper IP address of one or more DNS servers (use **cat /etc/resolv.conf** to verify) and after using **ping** to test connectivity to the DNS server.

Command	Description
dig	<p>A powerful tool for gathering information and testing name resolution. It is installed on most Linux distributions. Output is displayed in an answer section. Output will include the IP address mapped to the domain name, the DNS server that answered the query, and how long it took to receive that answer.</p> <p>The basic syntax is dig {domain name}</p> <p>The command dig @{IP address} {domain name} will resolve the domain name against the DNS server specified by the IP address.</p>

Command	Description
nslookup	A tool for gathering name resolution information and testing name resolution. It is available on most Linux distributions as well as Microsoft Windows. This command has a non-interactive mode, in which you can provide a domain name in a single command, and an interactive mode, in which you can issue the command by itself and then provide domain names on separate consecutive prompts. The syntax for non-interactive mode is <code>nslookup {domain name}</code>
host	Another, simple tool capable of gathering information and testing name resolution. It is installed on most Linux distributions. The basic syntax is <code>host {domain name}</code> The command <code>host {domain name} {IP address}</code> will resolve the domain name against the DNS server specified by the IP address.

THE ip COMMAND FOR TROUBLESHOOTING

The `ip` command can be used for troubleshooting as well as for network configuration. The first step in troubleshooting network connectivity is to verify all settings are correct. The `ip addr` command enables an administrator to ensure the configuration is accurate.

Examples of troubleshooting with `ip addr` include:

- Check the IP address configuration: If the `ip addr` command reports back an address in the link-local range, then the NIC is not configured with a legitimate IP address. The link-local range is 169.254.#.#. If the system is a DHCP client, then verify connectivity to the DHCP server.
- Check the status of the NIC: The `ip` command can be used to "up" or "down" a NIC (enable or disable it). If the NIC shows as down, it is disabled and not functional.
- Check the usage statistics of a NIC: Using `ip` with the `-S` option enables you to view connectivity statistics for the connection.

THE route COMMAND

It is possible to configure a Linux system to act as a router. The role of a router is to pass traffic from one network segment to another, based on the network ID of packets. In order to properly direct traffic to the appropriate subnet (and prevent traffic from getting to certain subnets, too), routing table entries are configured. This is not a common configuration for Linux hosts, but is important to be aware of.

The `route` command is used to view the routing table. The command is also used to manipulate the routing table, enabling the administrator to configure desired routes.

Examples include the following.

Command	Used To
<code>route</code>	View the current routing table on the system.

Command	Used To
route add default gw {IP address}	Configure a default gateway by its IP address. Packets will be passed to this destination if there are no other routes that match their network ID.
route add -host {IP address} reject	Filter traffic destined to the specified address, which enables an administrator to control connections to a particular host. Can also be configured for an entire subnet.



Note: The `route` command is considered deprecated in favor of the `ip route` command.

```
root@server01:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway        0.0.0.0        UG    100    0        0 enp3s0
10.50.1.0       0.0.0.0        255.255.255.0   U      100    0        0 enp3s0
192.168.122.0   0.0.0.0        255.255.255.0   U      0      0        0 virbr0
[root@server01 ~]#
```

Displaying the routing table.

SYNTAX

The syntax of the `route` command is `route [options]`

THE nmap COMMAND

Network Mapper, or `nmap`, is a powerful tool for exploring a network environment. It identifies nodes and is often able to report back available services, operating system versions, hostnames, IP addresses, MAC addresses, network devices (switches, routers), network printers, etc. The `nmap` utility has a great many options. It also has a GUI version called Zenmap.

The `nmap` utility may be used initially to audit and document the network. In troubleshooting, having such documentation is essential. It can also be used directly in the troubleshooting process to confirm whether expected components are in place or if there have been changes to the network environment.

The following is an example troubleshooting process:

1. `ip addr` —does the local host have the correct IP configuration?
2. `ping {destination}` —is traffic able to flow from the source to the destination and back?

3. **nmap**—view the network structure to verify the existence of a path between the source and destination systems.



Note: The first two commands in this process are applicable to most network troubleshooting situations, including those that use many of the other tools discussed in this topic.

```
[root@server01 ~]# nmap server02
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-08 20:09 GMT
Nmap scan report for server02 (10.50.1.102)
Host is up (0.00051s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
MAC Address: C8:60:00:33:C4:A9 (Asustek Computer)

Nmap done: 1 IP address (1 host up) scanned in 5.37 seconds
[root@server01 ~]#
```

Scanning a target host over the network.

SYNTAX

X

The syntax of the **nmap** command is **nmap [options] {target}**

TROUBLESHOOTING EXAMPLES

The following are some examples of troubleshooting with **nmap**:

- **nmap -p 1-65535 -sV -sS -T4 {target}** —Port scan for all listening ports on the designated target (hostname, IP address, subnet). This ensures the destination computer is listening for the source computer's connection attempt.
- **nmap -sP 192.168.1.0/24** —Host discovery scan for all devices on the 192.168.1.0/24 subnet. This reports all devices detected on the designated subnet.

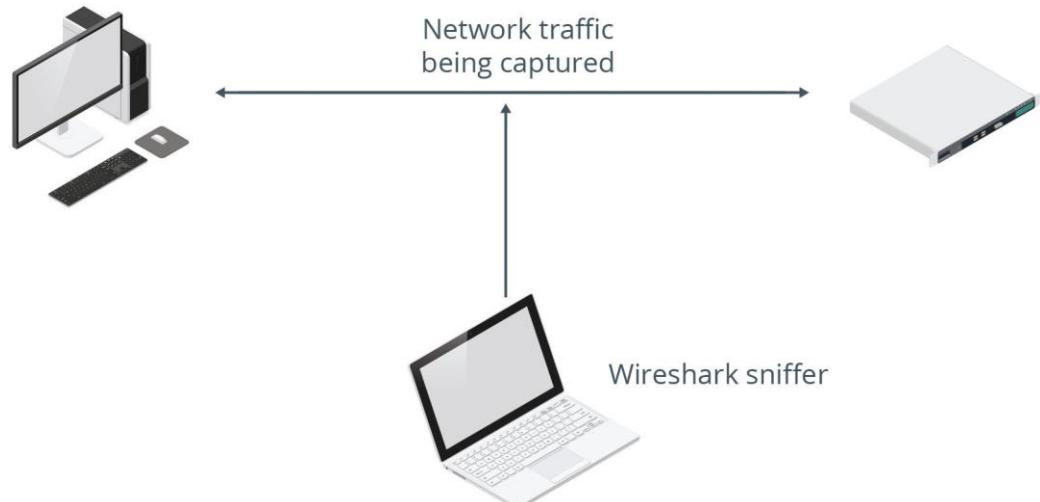
WIRESHARK

Wireshark is a very common packet sniffer and network analyzer. Network analyzers are used to intercept and potentially read network traffic. These tools may be used for eavesdropping attacks, but also for network troubleshooting. When network traffic is intercepted, information such as source/destination MAC address, source/destination IP address, port numbers, and packet payload (data) is exposed. One advantage of a tool like Wireshark is the ability to see exactly what packets are moving through a network segment or NIC and what packets are not. This is very useful for troubleshooting.

The following is an example troubleshooting process:

1. **ip addr** —does the local host have the correct IP configuration?
2. **ping {destination}** —is traffic able to flow from the source to the destination and back?

3. **firewall-cmd --list-services**—view what traffic may be filtered by the local firewall.
4. Wireshark—identify what network traffic is moving in a given network subnet.



Wireshark capturing network traffic as it flows between two hosts.



Note: Wireshark is primarily used as a GUI, but it does include the `tshark` command for CLI use.

THE `tcpdump` COMMAND

Another network analyzer is the `tcpdump` utility. Created in 1987, `tcpdump` remains one of the most popular packet sniffers available. It is installed by default on many Linux distributions. Users can determine traffic type and content using this command. It provides similar information to Wireshark, and you can use it in a similar troubleshooting process.

```

Captured packets
root@server01:~# tcpdump -i enp3s0 -c 5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp3s0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:16:48.093656 IP server01.42756 > a23-43-62-169.deploy.static.akamaitechnologies.com
.http: Flags [.], ack 2239704302, win 245, options [nop,nop,TS val 523852448 ecr 18045
16054], length 0
20:16:48.094371 IP server01.44454 > google-public-dns-a.google.com.domain: 50842+ PTR?
169.62.43.23.in-addr.arpa. (43)
20:16:48.096209 IP a23-43-62-169.deploy.static.akamaitechnologies.com.http > server01.
42756: Flags [.], ack 1, win 130, options [nop,nop,TS val 1804526071 ecr 523812369], l
ength 0
20:16:48.205388 IP google-public-dns-a.google.com.domain > server01.44454: 50842 1/0/0
PTR a23-43-62-169.deploy.static.akamaitechnologies.com. (107)
20:16:48.205650 IP server01.51317 > google-public-dns-a.google.com.domain: 49853+ PTR?
101.1.50.10.in-addr.arpa. (42)
5 packets captured
8 packets received by filter
0 packets dropped by kernel
[root@server01 ~]#
  
```

Summary

Capturing network packets.

Some common options with `tcpdump` are provided in the following table.

Option	Used To
<code>-i</code>	Specify the interface to use.
<code>-n</code>	Not resolve hostnames, speeding up the capture.
<code>-v</code>	Specify verbose mode.

SYNTAX

The syntax of the `tcpdump` command is `tcpdump [options] [-i {interface}] [host {IP address}]`

THE netcat COMMAND

The `netcat` command can be used to test connectivity and send data across network connections. The command may be spelled out as "netcat" or abbreviated as "nc" depending on the distribution. Systems may be identified by IP address or by hostname.

When troubleshooting, use `netcat` to listen on the destination computer and attempt a connection from the source computer in order to verify network functionality.

The following table provides some example use cases for the `netcat` command.

Use Case	Command Example
Connect two computers for the purpose of transferring information	On comp1 (listen on port): <code>netcat -l 4242</code> On comp2 (connect to listener): <code>netcat comp1 4242</code>
Transfer file content between two computers	On comp1 (listen on port): <code>netcat -l 4242 > received.file</code> On comp2 (connect to listener): <code>netcat comp1 < original.file</code>
Portscan a computer	<code>netcat -z -v domain.tld 1-1000</code> (scans ports 1 to 1000).

 **Note:** TLD stands for top-level domain, and can include .com, .net, .org, etc.

SYNTAX

The syntax of the `netcat` command is `netcat [options]`

THE iftop COMMAND

The `iftop` command displays bandwidth usage information for the system, helping to identify whether a particular NIC or protocol is consuming the most bandwidth. The `iftop` command may not be installed on all Linux distributions.

This command can help you identify why a particular link may be slow by showing the traffic on that connection. You can use it to check to see what is consuming the most bandwidth on an interface. For example: `iftop -i eth0`

Network slowness is often a symptom of bandwidth saturation, in which a network link's capacity is exceeded, i.e., all bandwidth is being used up. This can lead to degraded network performance or even service outages. With the **iftop** command, you can investigate any NICs on a network link that you suspect may be sending or receiving excessive sums of traffic across that link. For example, one host might be making repeated requests to an internal web server, and both hosts might be flooding the network with their requests and responses. Once you've identified the source of the issue, you can then take steps to stop the offending host from making these requests, such as terminating the service responsible for the requests.

The screenshot shows the output of the **iftop** command on a terminal window titled "root@server01:~". The output is divided into two main sections: "Connection stats" and "Summary".

Connection stats:

	12.5Kb	25.0Kb	37.5Kb	50.0Kb	62.5Kb
server01	=> google-public-dns-a.g	0b	59b	126b	
	<=	0b	83b	159b	
server01	=> mirror.siena.edu	0b	42b	23b	
	<=	0b	0b	0b	
server01	=> 192.168.37.155	0b	0b	23b	
	<=	0b	0b	18b	

Summary:

TX:	cum:	387B	peak:	1rates:	0b	101b	172b
RX:		398B		1.15Kb	0b	83b	177b
TOTAL:		785B		2.16Kb	0b	184b	349b

Displaying connection bandwidth statistics.

SYNTAX

The syntax of the **iftop** command is **iftop [options] [-i {interface}]**

THE iperf COMMAND

The **iperf** command is used to test the maximum throughput an interface will support. The utility must be installed on both endpoint systems. One system is designated as a "server" and the other as a "client." It is the **iperf** client that is getting tested. You can use this command to ensure that throughput is meeting your expectations.

A basic test is as follows:

1. On the server, run **iperf -s**
2. On the client, run **iperf -c {server address}**
3. Examine the results that appear.

The screenshot shows a terminal window with the following output:

```

root@server01 ~]# iperf -c server02
Client connecting to server02, TCP port 5001
TCP window size: 85.0 KByte (default)
[  3] local 10.50.1.101 port 54382 connected with 10.50.1.102 port 5001
[ ID] Interval      Transfer     Bandwidth
[  3]  0.0-10.0 sec   112 MBytes   94.3 Mbits/sec

```

Annotations with rounded blue boxes:

- A box labeled "Remote host" points to the command `iperf -c server02`.
- A box labeled "Test results" points to the bandwidth measurement line "[3] 0.0-10.0 sec 112 MBytes 94.3 Mbits/sec".

Testing network bandwidth between two hosts.

SYNTA X

The syntax of the **iperf** command is **iperf {-c|-s} [options]**

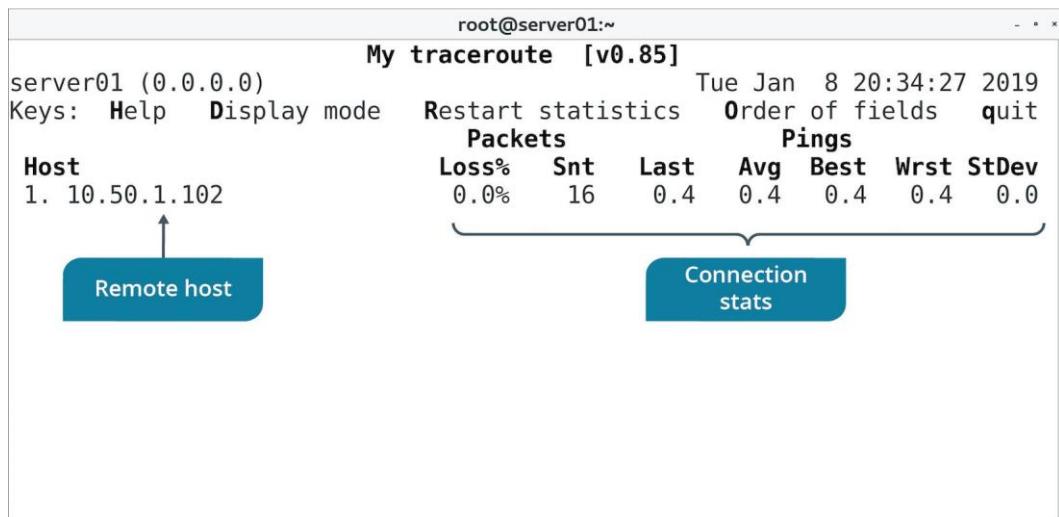
THROUGHPUT VS. BANDWIDTH

Bandwidth is the potential amount of data that may move through a network connection in a given amount of time. Throughput is the amount of data that actually moves through a network connection in the given amount of time. Both the **iftop** and **iperf** utilities measure throughput.

THE mtr COMMAND

The **mtr** utility is a combination of **ping** and **traceroute**, with additional improvements to enable testing of the quality of a network connection. Ping packets are sent to the destination in large groups, with **mtr** noting how long responses take to the packets.

The **mtr** command also takes note of lost packets, a symptom of a problem called packet drop or packet loss. This occurs when one or more packets sent from a source are unable to reach their intended destination. Packet loss can cause latency if the packets are queued for retransmission, or the data may not be successfully transmitted at all. A large number of lost packets are a strong indicator of a network issue along the path. By identifying that the issue exists, as well as where in the path it exists, **mtr** enables an administrator to find potentially failed networking components. The output of **mtr** identifies the percentage of packets along the path that are dropped, and one or more nodes in that path experiencing a high percentage of packet loss may be at fault.



Analyzing connection information to a remote host.

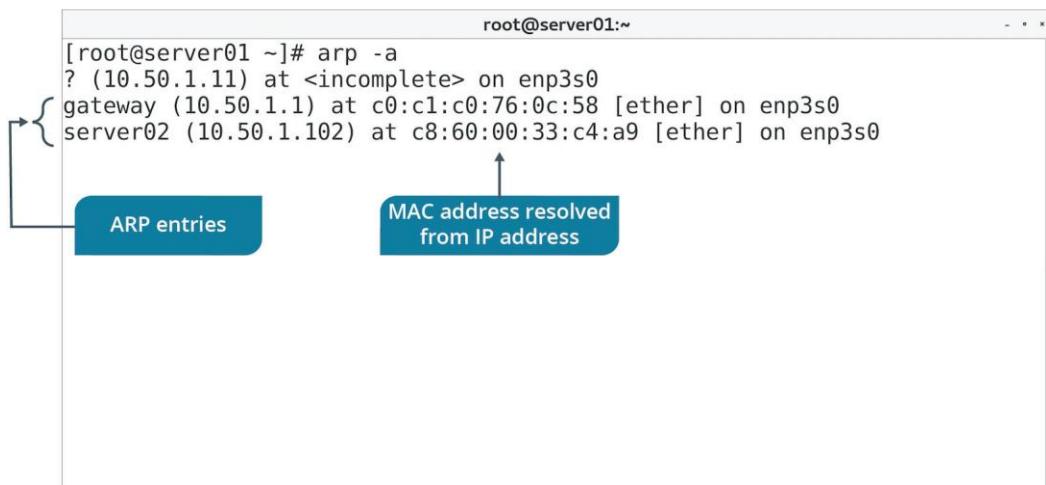
SYNTAX

The syntax of the **mtr** command is **mtr [options] [hostname]**

THE arp COMMAND

As you know, nodes on the network typically have three identities: hostname, IP addresses, and MAC addresses. DNS translates hostnames to IP addresses. The **Address Resolution Protocol (ARP)** is used to relate IP addresses and MAC addresses. There is also an **arp** command that administrators can run to discover information about known MAC addresses.

Computers will cache recently resolved MAC and IP address combinations. If a computer has cached incorrect or out-of-date information, connectivity may be lost to a particular node. The ARP cache can be cleared as part of the troubleshooting process. For example, you can run **arp -d {IP address}** to clear an entry for a particular IP address, and then try to ping the host again. Use **arp -a** to view the cache.



The ARP cache.

SYNTAX

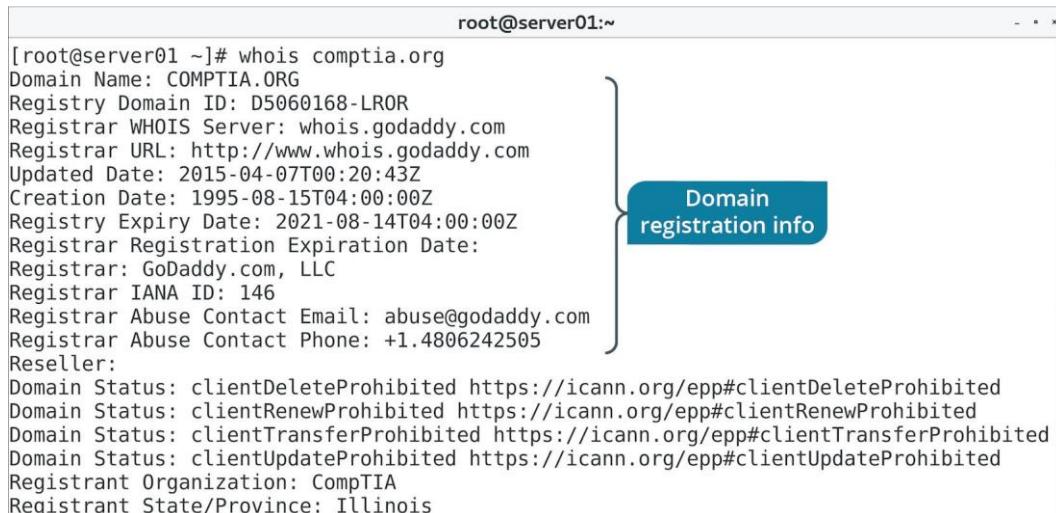
The syntax of the **arp** command is **arp [options]**

THE whois COMMAND

The **whois** command provides information on Internet DNS registrations for organizations. This can be useful for learning or verifying information regarding ownership of a domain name, contact information for an organization, etc.

Some examples include:

- **whois google.com**
- **whois ubuntu.com**
- **whois redhat.com**



```
root@server01 ~]# whois comptia.org
Domain Name: COMPTIA.ORG
Registry Domain ID: D5060168-LR0R
Registrar WHOIS Server: whois.godaddy.com
Registrar URL: http://www.whois.godaddy.com
Updated Date: 2015-04-07T00:20:43Z
Creation Date: 1995-08-15T04:00:00Z
Registry Expiry Date: 2021-08-14T04:00:00Z
Registrar Registration Expiration Date:
Registrar: GoDaddy.com, LLC
Registrar IANA ID: 146
Registrar Abuse Contact Email: abuse@godaddy.com
Registrar Abuse Contact Phone: +1.4806242505
Reseller:
Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited
Domain Status: clientRenewProhibited https://icann.org/epp#clientRenewProhibited
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited
Registrant Organization: CompTIA
Registrant State/Province: Illinois
```

Looking up registration information for the public CompTIA domain.

SYNTA

X

The syntax of the **whois** command is **whois [options] {domain name}**

GUIDELINES FOR TROUBLESHOOTING NETWORKING ISSUES

Use the following guidelines when troubleshooting networking issues.

TROUBLESHOOT NETWORKING ISSUES

When troubleshooting networking issues:

- Narrow the scope of the problem—if the issue occurs on one machine, it's likely a misconfiguration on that machine. If the issue occurs on several machines, it could be a network problem. If several nodes are affected, check to see what they have in common. Are they all connected to the same switch or wireless access point? Are they all in the same subnet?
- Verify the IP address configuration on the system using the **ip** command. If the system has an IP address beginning with 169.254.x.x, then it failed to lease an IP address configuration from a DHCP server, which gives you a place to start.
 - Attempt to renew the dynamic IP address configuration from a DHCP server.
 - Verify there are no typographical errors if the IP address configuration is static.

- Use the following tools and processes to diagnose bandwidth issues when experiencing slow network performance:
 - Continuously ping affected hosts to confirm that a connection between hosts is slow.
 - Use **traceroute** to identify where along a routing path the network bottleneck might be occurring.
 - Use **iftop** on affected or suspected hosts to identify what applications are consuming bandwidth. Some applications, like streaming media, can consume a great deal of bandwidth if left unchecked.
- Use **iperf** to test network interface throughput if a connection is slow, but bandwidth is not the issue. The interface may itself be unable to handle the desired speeds and require an upgrade or replacement.
- Utilize these general commands:
 - Use the **ping** command to test connectivity.
 - Use the **traceroute** command to understand the network between two endpoints.
 - Use the **nslookup** and **host** commands to test name resolution.
 - Use the **ethtool** command to verify the network card driver and configuration information.
 - Use the **virsh** subcommands to verify and manage KVM virtual machines.
 - Use the **iperf** tool to test throughput between two endpoints.

Activity 10-10

Discussing Networking Issues

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You're asked to design the network service implementation at a branch office location. There will be servers, printers, and workstations in the office. Which approach for assigning IP addresses to workstations would you take?

2. Your team has decided to use only one internal DNS server and one external. You must change the DNS server IP addresses on every Linux server. Which file can you edit to change the IP addresses?

3. Why do we use a service such as DNS to map IP addresses to names?

4. You're assisting a group of web developers to set up their test systems, but you don't want other users to know about them, so you aren't registering them with your network DNS servers. Instead, you're helping the developers edit a local file on their systems that will act as a local IP address-to-name mapper. Which file can you edit for this private name resolution service?

5. Your team is setting a standard for the /etc/nsswitch.conf file. After some debate, the team decides to use the preferred configuration for the ordering of name lookup services. What order is this?

Activity 10-11

Testing the Network Environment

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You will work with a partner in this activity.

SCENARIO

You want to use some of the Linux network troubleshooting utilities so that you can better understand the Develetech network environment. These will help you diagnose and solve issues related to latency, lack of hostname resolution, inability to connect to other hosts, and more.

1. View network services that are currently listening on the hosts in your network.
 - a) Enter **ip addr** to verify the system has a correct IP address configuration.


Note: When troubleshooting, an IP address that begins with 169.254 indicates the client could not lease an IP address from a DHCP server.

 - b) Enter **ss -l | less** to see what TCP ports your system is currently listening on, then press **q** to return to the prompt.
 - c) Enter **nc localhost 21**
You should receive a "Connection refused" error, indicating that your system is not listening on port 21 (FTP).
 - d) Enter **nc <partner hostname> 22** to verify that your partner's host is listening on port 22 (SSH).
 - e) Press **Ctrl+C** to disconnect.
You can use a tool like **nc** to identify network services that aren't listening on the local or remote host.
2. Test public name resolution.
 - a) Enter **host www.comptia.org**
 - b) Verify that you resolved the public CompTIA hostname to a specific IP address.
You can use a name resolution tool like **host** to ensure that you can establish a connection to hosts using human-friendly hostnames.
3. Capture network traffic.
 - a) Enter **sudo tcpdump -i <device ID>** where **<device ID>** is your Ethernet device name.
 - b) Verify that the tool is listening on the device.
 - c) Right-click the desktop and select **Open Terminal** to open another terminal.
 - d) In this new terminal, enter **ping <partner hostname> -c 4**

- e) In the other terminal window, verify that **tcpdump** captured the ICMP echo traffic.

```
20:45:52.678323 IP server02 > server01: ICMP echo request, id 27664, seq 1, length 64
20:45:52.678377 ARP, Request who-has server02 tell server01, length 28
20:45:52.678590 ARP, Reply server02 is-at c8:60:00:33:c4:a9 (oui Unknown), length 46
20:45:52.678594 IP server01 > server02: ICMP echo reply, id 27664, seq 1, length 64
20:45:53.678387 IP server02 > server01: ICMP echo request, id 27664, seq 2, length 64
20:45:53.678426 IP server01 > server02: ICMP echo reply, id 27664, seq 2, length 64
20:45:54.678387 IP server02 > server01: ICMP echo request, id 27664, seq 3, length 64
20:45:54.678420 IP server01 > server02: ICMP echo reply, id 27664, seq 3, length 64
20:45:55.678380 IP server02 > server01: ICMP echo request, id 27664, seq 4, length 64
20:45:55.678419 IP server01 > server02: ICMP echo reply, id 27664, seq 4, length 64
```

You can use a network capture tool like **tcpdump** to learn more about the traffic that is transmitted and received over your network.

- f) Close the terminal window running the **tcpdump** capture.
-

Activity 10-12

Troubleshooting Networking Issues

SCENARIO

The Develetech server team has collected a list of common troubleshooting issues, based on help desk tickets. They have asked you to consider these scenarios and suggest Linux utilities and procedures to make troubleshooting more effective and efficient.

1. Rose Stanley calls the help desk, registering a ticket that her Linux workstation does not have network connectivity. What process and tools might you suggest?
 2. A change has been made to the configuration of a router by a member of the network team. Unfortunately, the change was not documented and the team member is unavailable. You have no access to the router itself. You would like to investigate what network traffic is moving on each side of the router to help determine what ports might have been closed. What tools might you use to gather network traffic on each side of the router?
 3. What steps might you take to implement a plan of action regarding the gathering of network traffic?
 4. The Develetech IT staff recognizes the importance of solid documentation and wants to generate a representation of the entire network. The staff wants the information to include all routers and servers, as well as listening services and operating system information for each server. What tool might you suggest for this project?

5. A junior Develetech server administrator believes that the bandwidth usage on his Linux server is being consumed by a particular network service. What tool could you suggest to him to gather information about exactly what protocols are using bandwidth on the NIC?

 6. After the administrator gathers the desired information, he discovers that there is no unexpected bandwidth usage by the services. He wonders if perhaps there is an issue on the network itself. He asks for a utility that would enable him to empirically test network bandwidth between two servers. What might you suggest?

 7. One of Develetech's server administrators is having difficulties with name resolution. She is responsible for both Linux and Windows servers. She wants to know if there is a single tool she can run from both of her servers to confirm they are receiving name resolution information from a specific DNS server.
-

Summary

In this lesson, you added Linux systems to your network and configured and managed various network services. By making your systems network-aware and configuring them for various network roles, you can facilitate communication between many other computers across a LAN or the Internet.

What kind of network services does your organization provide to employees and customers?

Does your organization leverage cloud or virtualization technologies? If so, what resources use these technologies? If not, do you see these technologies being a benefit to the organization?



Practice Question: Additional practice questions are available on the course website.

Lesson 11

Managing Packages and Software

LESSON TIME: 2 HOURS, 30 MINUTES

LESSON INTRODUCTION

Unlike in other operating systems, software is provided to Linux® systems in multiple different ways. You need be able to acquire and manage software packages in order to install the necessary applications and keep them updated. Since packaged software is typically delivered over a network like the Internet, it'll be easier for you to manage those packages now that you've configured networking on your Linux systems.

LESSON OBJECTIVES

In this lesson, you will:

- Identify the most common package managers in Linux, including RPM and dpkg.
- Manage RPM packages with the YUM front-end.
- Manage Debian packages with the APT front-end.
- Configure package repositories.
- Acquire software through means other than package managers.
- Compile software packages that are in source code form.
- Troubleshoot issues with software dependencies.

Topic A

Identify Package Managers



EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

There are multiple utilities that enable you to manage packages on your Linux systems. In this topic, you'll explore the major package managers that are available.

PACKAGE MANAGERS

Linux distributions rely on two different methods of managing the software lifecycle. The first method is **package managers**—programs that install, update, inventory, and uninstall packaged software. The second method is compiling software manually from source code. The open source nature of Linux means that compiling code is much more common for Linux administrators than for Windows or macOS users.

Package managers govern the software lifecycle, making it much easier for Linux administrators to control what software is installed, manage software versions, and to uninstall the software. The term *package* refers to a collection of files needed for a particular program. This set of files includes the pre-compiled application itself, any supporting files it might require, and supporting documentation. Packages are easy to distribute, verify, and manage via package managers.

SOFTWARE DEPENDENCIES

Many Linux applications are modular and depend on other pieces of software already being present. The packages will list dependencies, the required components without which the application cannot function properly. Package managers will check for these dependencies before installing the software from the package. A "failed dependency" error indicates that one or more of these dependencies has not been satisfied.

RED HAT INSTALLATION VS. DEBIAN INSTALLATION VS. COMPILING

There are two dominant methods for managing software packages. The first method, created by Red Hat® in 1995, is called the **Red Hat Package Manager (RPM)**. The second method, created in 1994, is the Debian **dpkg** system. The two managers are significantly different from each other but functionally provide the same end result. As a Linux user, your choice will usually be driven by your preferred Linux distribution.

The vast majority of Linux distributions trace their origins back to either Red Hat Linux or Debian Linux.

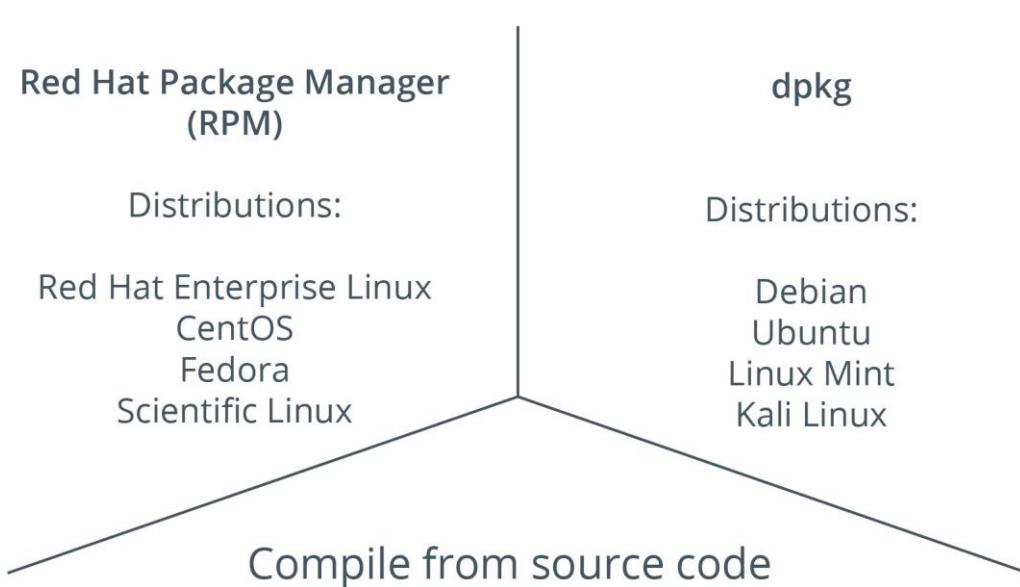
An alternative to using package managers is to **compile** the software yourself. This is the traditional method of managing software. It has advantages and disadvantages. Windows® and macOS® users don't typically consider compiling software, because most of the software available is closed source, and therefore is only available in a pre-compiled format. Because Linux relies on open source software, access to the source code is much more common. Users can make changes to the source code, enabling or disabling options, customizing installation choices, and even expanding the functionality of the software before compiling the code on their systems.

Manually compiling software requires more effort and more knowledge, but it can result in a more useful piece of software.

DISTRIBUTION ORIGINS

The following table lists the origins of several common Linux distributions.

Red Hat Linux	Debian Linux
Red Hat® Enterprise Linux® (RHEL)	Debian
CentOS®	Ubuntu®
Fedora®	Linux Mint
Scientific Linux	Kali Linux
VMware ESXi™	Raspbian
Amazon™ Linux	SteamOS
Oracle® Linux	openSUSE



The primary software installation methods for Linux distributions.

RED HAT PACKAGE MANAGERS

Red Hat's package manager is called RPM. It is still in use today by many administrators. Software packages that are prepared for RPM use the .rpm file extension. RPM is very flexible and very powerful. One of its most useful features is the ability to inventory software, enabling administrators to easily see installed software and software version information.

There is a newer and more advanced package manager that is commonly used by Red Hat derivatives. This package manager is called the **Yellowdog Updater, Modified (YUM)**. It relies on RPM and uses .rpm packages. It offers a more elegant set of commands and greater flexibility for using software repositories and handling dependencies. Today, even Red Hat prefers YUM to manage the software lifecycle.

DEBIAN PACKAGE MANAGERS

Debian's package manager is called dpkg. It is still available for use in Debian and Debian-derivatives. The dpkg manager can control the entire software lifecycle. Software packages with the .deb file extension can be managed using dpkg.

The preferred method of package management in Debian-derivatives today is the **Advanced Package Tool (APT)**. It is more flexible than the original dpkg package manager. It also relies on .deb packages. Most software management tasks on Debian-derived distributions will use APT.

DNF AND ZYPPER

The **Dandified YUM (DNF)** package manager is an improved version of YUM. It uses fewer resources while still maintaining support for the fundamental RPM package manager. It includes a simplified set of commands as compared to YUM. Most of the YUM subcommands are used by DNF, so it will be relatively familiar to those used to using YUM.

Zypper is a package manager that supports repositories, dependency solving, and management of the software lifecycle. Zypper is an openSUSE package manager that supports .rpm packages. It is very efficient and does an excellent job of managing package dependencies.

DNF SYNTAX

The syntax to install a DNF package is `dnf install {package name}`

The syntax to uninstall a DNF package is `dnf remove {package name}`

ZYPPER SYNTAX

The syntax to install a Zypper package is `zypper in {package name}`

The syntax to uninstall a Zypper package is `zypper rm {package name}`

Activity 11-1

Discussing Package Managers

SCENARIO

Answer the following questions to check your understanding of the topic.

1. Your company has standardized on CentOS as the corporate Linux distribution. You want the greatest amount of control over how programs are installed, where they are installed, and all install options. Which install method works best for your requirements?

2. You need to install the Apache web service on your new CentOS web server farm and you cannot decide between using a package manager and compiling. You look at both options thoroughly and decide that, for your current needs, installing via package manager will be adequate. Which package manager do you use to install Apache?

3. Your development team at Develetech needs to modify some MySQL database software to optimize it for a database application. You have installed MySQL using YUM. What do you need to do to accommodate the modification need?

4. What is the purpose behind using a package manager to install software?

5. You have used Red Hat Enterprise Linux and some of its derivative distributions, but you want to try a Debian-based distribution because of the APT package manager. Aside from Debian itself, what are some common Debian-based distributions you can choose from?

Activity 11-2

Identifying Package Managers

SCENARIO

Develetech will be using both Ubuntu and CentOS servers in its infrastructure. It is up to you to design and configure a software management solution. First, you will decide what solutions are appropriate.

1. **What is the common package manager and package management utility for Red Hat-derived distributions?**

2. **What is the common package manager and package management utility for Debian-derived distributions?**

3. Four of Develetech's server and development systems are listed below. Identify which package manager is appropriate for each of them.

CentOS Web Server

Ubuntu Development

Workstation Raspberry Pi with

Raspbian Linux Scientific Linux

4. **Why might you choose to compile software from source code as opposed to using a pre-compiled package?**

Topic B

Manage RPM Packages with YUM



EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Because RPM packages are central to Red Hat, one of the most prominent distribution families, you'll use several tools to manage those types of packages.

THE rpm COMMAND

The **rpm** command is used to manage RPM packages on Red Hat-derived distributions. It includes many different options, some of which are described in the following table.

Option	Used To
-i {package name}	Install the specified software.
-e {package name}	Erase (uninstall) the package.
-v	Enable verbose mode, providing more detail.
-h	Print hash marks to indicate a progress bar.
-V {package name}	Verify the software components of the package exist.

```
root@server01 ~]# rpm -vV iperf
..... /usr/bin/iperf
..... /usr/share/doc/iperf-2.0.12
..... d /usr/share/doc/iperf-2.0.12/AUTHORS
..... d /usr/share/doc/iperf-2.0.12/COPYING
..... d /usr/share/doc/iperf-2.0.12/ChangeLog
..... d /usr/share/doc/iperf-2.0.12/README
..... d /usr/share/doc/iperf-2.0.12/dast.gif
..... d /usr/share/doc/iperf-2.0.12/index.html
..... d /usr/share/doc/iperf-2.0.12/ui_license.html
..... d /usr/share/man/man1/iperf.1.gz
[root@server01 ~]#
```

Verifying the software components of a package.

SYNTAX

The syntax of the **rpm** command is **rpm [options] [package name]**

RPM QUERYING

One of the most powerful features of the RPM package manager is the ability to maintain a database of software information. This database enables administrators to

discover package version information, list all installed software, discover dependency information, etc. Example command queries include the following.

Command	Used To
rpm -qa	List all installed software (typically a very large output).
rpm -qi {package name}	List information about a particular package.
rpm -qc {package name}	List the configuration files for a particular package.

```
[root@server01 ~]# rpm -qi iperf
Name        : iperf
Version     : 2.0.12
Release    : 4.el7
Architecture: x86_64
Install Date: Tue 08 Jan 2019 08:26:20 PM GMT
Group       : Applications/Internet
Size        : 181567
License     : BSD
Signature   : RSA/SHA256, Wed 11 Jul 2018 02:44:18 PM BST, Key ID 6a2faea
2352c64e5
Source RPM  : iperf-2.0.12-4.el7.src.rpm
Build Date  : Wed 11 Jul 2018 02:35:39 PM BST
Build Host  : buildvms-03.phx2.fedoraproject.org
Relocations : (not relocatable)
Packager    : Fedora Project
Vendor      : Fedora Project
```

Querying information about a package.

RPM VERIFICATION

RPM can be used to verify software. The verification will check to see if the installed software components match what the RPM package specifies should be installed. The verify option is used when troubleshooting installed software to ensure that the entire software package is installed and valid.

RPM UPGRADES

Part of the software lifecycle is to keep software current. RPM offers two primary ways of accomplishing this goal:

- **-U** —upgrades the installed package, and installs the package if it is not already installed.
- **-F** —freshens the installed package, i.e., upgrades but does not install the package if it is not already installed.

THE yum COMMAND

The **yum** command improves on the functionality of **rpm** while still using .rpm packages and maintaining an RPM database. It provides a more straightforward method for managing packages.

One of the biggest benefits of YUM is the ability to automatically handle software dependencies. This means that administrators can tell YUM to install a particular

package, along with automatically installing any additional packages that package depends on.

An additional YUM benefit is the use of repositories. Repositories are storage locations for .rpm files. Repositories enable administrators to more easily maintain version control over software.

```
[root@server01 ~]# yum install iperf3
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.cc.columbia.edu
 * epel: fedora-epel.mirrors.tds.net
 * extras: mirror.cc.columbia.edu
 * updates: mirror.cc.columbia.edu
Resolving Dependencies
--> Running transaction check
--> Package iperf3.x86_64 0:3.1.7-2.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
| Package      | Arch | Version | Repository | Size |
|=====|
| Installing: |       |          |            |       |
| iperf3       | x86_64 | 3.1.7-2.el7 | base      | 79 k |
```

Installing a software package using YUM.

yum SUBCOMMANDS

The yum command comes with several subcommands for managing packages.

Subcommand	Used To
<code>install {package name}</code>	Install the package from any configured repository.
<code>localinstall {package name}</code>	Install the package from the local repository.
<code>remove {package name}</code>	Uninstall the package.
<code>update [package name]</code>	Update the package; if none provided, updates all installed packages (time-consuming).
<code>info {package name}</code>	Report information about the package.
<code>provides {file name}</code>	Report what package provides the specified files or libraries.

SYNTAX

The syntax of the yum command is `yum [options] [subcommand] [package name]`

THE -y OPTION

Use the `-y` option with the yum command to automatically answer yes to installing additional software dependencies. If you do not, YUM will prompt you to answer yes or no to whether the additional dependencies should be installed.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 11-3

Discussing Managing Software with RPM and YUM

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You are teaching a junior administrator how to install software via YUM. You ask him to install the Lynx text-based web browser (`lynx`) on a test system. What should the install command look like?

2. You need to remove the Apache web service software package (`httpd`) so that you can install via source code. Using YUM, what command can you issue to uninstall Apache?

3. You want to install the `xterm` package onto your Linux workstation and don't want to wait for `xterm` and all its dependencies to be found before you acknowledge the installation. Using YUM, how can you install `xterm` and its dependencies without having to interact with the installation?

4. You need to install several packages that the developers have downloaded and placed into a shared directory (`/scratch`). These are all RPM packages. How can you install the `rdesktop-0.9.rpm` package, for example?

5. You have taken a position as a system administrator at a new company and have done some discovery work on the environment. You've noticed that multiple software packages are more than one year out of date on several CentOS systems. Using YUM, which command can you issue to update all software packages on each system without acknowledging or confirming the update?

Activity 11-4

Managing Software with RPM and YUM

DATA FILES

/Packages/ksh-20120801-137.el7.x86_64.rpm

/Packages/vsftpd-3.0.2-22.el7.x86_64.rpm

BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

SCENARIO

One of the other Develetech administrators has asked you to demonstrate the software management lifecycle on a CentOS server. You will use the KornShell (**ksh**) as an example of how to use the **rpm** command, and then the Very Secure FTP daemon (**vsftpd**) to demonstrate the **yum** command.

1. Use the **rpm** command to manage the software lifecycle.
 - a) If necessary, open a terminal.
 - b) Enter **sudo rpm -ivh /Packages/ksh-20120801-137.el7.x86_64.rpm** to install the **ksh** package in verbose mode and with a hash progress bar.
 - c) Enter **rpm -qi ksh** to view information on the **ksh** package.
 - d) Enter **sudo rpm -Vv ksh** to verify the **ksh** installation.
 - e) Enter **sudo rpm -qI ksh** to list the files in the **ksh** package.
 - f) Enter **sudo rpm -e ksh** to "erase" or uninstall the **ksh** package.

 2. Use the **yum** command to manage the software lifecycle.
 - a) Enter **yum info vsftpd** to discover information about the **vsftpd** package.
 - b) Enter **sudo yum localinstall /Packages/vsftpd-3.0.2-22.el7.x86_64.rpm** to install the **vsftpd** package.
 - c) Enter **Y** when prompted to complete the installation.
- 

Note: If you include a **-y** option with **yum**, it will automatically answer yes to this prompt and not pause the installation.
- d) Enter **yum info vsftpd** to view information on the **vsftpd** package.
 - e) Enter **yum provides /etc/vsftpd/vsftpd.conf** to discover what package the configuration file belongs to.
 - f) Enter **sudo yum -y remove vsftpd** to uninstall the **vsftpd** package.

Topic C

Manage Debian Packages with APT



EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

The other prominent distribution family with its own package manager is Debian. In this topic, you'll use various tools to manage Debian packages.

THE `dpkg` COMMAND

The `dpkg` command is used to manage packages on Debian-derived distributions. It includes many different options, some of which are described in the following table.

Option	Used To
<code>-i {package name}</code>	Install the package.
<code>-r {package name}</code>	Remove (uninstall) the package.
<code>-l [package name]</code>	List information about the specified package; if none provided, list all installed packages.
<code>-s {package name}</code>	Report whether the package is installed.

SYNTAX

The syntax of the `dpkg` command is `dpkg [options] [package name]`

DEBIAN PACKAGE VERIFICATION

The `-i` installation option with the `dpkg` command repairs a software installation by ensuring all necessary components are installed.

THE `apt` COMMAND

Although `dpkg` is the original installer for Debian-derived distributions, today `.deb` packages are more commonly managed using APT. APT is a front-end manager to the `dpkg` system, much like YUM is a front-end manager to the RPM system.

Until recently, the common software management tools were implemented as a mix of the `apt-get` and the `apt-cache` commands, along with several other variations on the `apt-*` format. Many Debian-derived distributions now use the more streamlined package manager simply named `apt`.

apt SUBCOMMANDS

The `apt` command comes with several subcommands for managing packages.

Subcommand	Used To
<code>install {package name}</code>	Install the package.
<code>remove {package name}</code>	Uninstall the package, leaving behind its configuration files.

Subcommand	Used To
purge {package name}	Uninstall the package and remove its configuration files.
show {package name}	Report information about the package.
version {package name}	Display version information about the package.
update	Update APT database of available packages.
upgrade [package name]	Upgrade the package, or upgrade all packages if none provided (time-consuming).

SYNTAX

The syntax of the `apt` command is `apt [options] [subcommand] [package name]`

THE apt-get AND apt-cache COMMANDS

The `apt-get` and `apt-cache` commands are still functional, as they provide lower-level functionality and more specific controls than the `apt` command does, but they may be more confusing due to their large number of options and less organized command structure.

Most of the subcommands mentioned previously with `apt` have the same effect with the `apt-get` command. You can use `apt-cache show {package name}` to display package information.

DEBIAN PACKAGE UPGRADES

When using `apt` or `apt-get` to manage packages on your Debian-derived distribution, there are two subcommands that you must understand. The first is `update` and the second is `upgrade`.

The `apt update` command updates the APT database of available packages, enabling APT to become aware of new versions of software available in the repositories. This does not install any software. The `apt upgrade` command upgrades all installed software based on newer versions of the packages as seen by the APT database. This is a full upgrade of all software, and as such can be time-consuming. The `apt upgrade {package name}` command upgrades the specified package based on a newer version of the package as seen by the APT database.

It is important to run the `apt update` command first, and then run the `apt upgrade` command. If the `apt update` command is not run first, the database will not be aware of newer packages.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 11-5

Discussing Managing Software with dpkg and APT

SCENARIO

Answer the following questions to check your understanding of the topic.

1. You are teaching a CentOS Linux administrator how to install software on a Debian-based Linux system via APT. Which command can you use to install the text-based browser Lynx (lynx)?

2. You need to remove the MySQL software package (mysql) so that you can install the program via source code, but you want to keep all configuration files intact. Using APT, what command can you issue to uninstall MySQL while leaving the configuration files?

3. It has been a few months since the old system administrators, who are now no longer with the company, updated the Linux systems. The first thing you need to do is to update the APT database with a fresh list of available packages. How do you update the APT database?

4. You need to totally remove the emacs package and its configuration files as well. Using APT, what command can you use to do this?

5. You have taken a position as system administrator at a new company and have done some discovery work on the environment. You've noticed that multiple software packages are more than one year out of date on several Debian systems. Which command can you issue to update all software packages on each system?

Activity 11-6

Managing Software with dpkg and APT

BEFORE YOU BEGIN

You are logged in to the GUI as your student account. An Ubuntu VM has been prepared for you to use.

SCENARIO

Some Linux systems in Develetech run Ubuntu and other versions of Debian. Just like your Red Hat-based systems, these need to undergo network troubleshooting from time-to-time. So, you'll download and install the `nmap` package on these machines to ensure you have the right toolset for the job. You'll use `dpkg` and APT.

1. Load the Ubuntu VM.
 - a) From the desktop menu, select **Applications**→**System Tools**→**Virtual Machine Manager**.
 - b) Enter the root password.
 - c) Right-click **ubuntu-vm** and select **Run**.
 - d) Right-click **ubuntu-vm** and select **Open**.
 - e) Wait for the VM to load.
 - f) Select the **student** account, then enter **Pa22w0rd** as the password.

2. Update the APT database with current package version information.

- a) Right-click the desktop and select **Open Terminal**.
- b) Enter **sudo apt update**
- c) Enter the password when prompted.



Note: If the Ubuntu VM has no Internet connectivity, you may need to restart your CentOS host and then reload the VM.

- d) Verify that the database update operation completed.
You're presented with the number of packages that can be upgraded. If you wanted to upgrade an existing package, you could use the **apt upgrade {package name}** command. For now, you'll install a new package.

3. Download and install the `nmap` package.

- a) Enter **sudo apt install nmap**



Note: If you receive a "Could not get lock..." error, it means the APT package manager is automatically checking for updates. You could wait a few moments or kill the process manually.

- b) Enter **Y** to confirm the operation.



Note: You might need to maximize the VM window or scroll to see the prompt asking you to confirm the operation.

- c) Wait for the installation to complete.
 - d) Enter `apt show nmap` to discover information about the `nmap` package.
 - e) Enter `nmap localhost` to test the utility, confirming that it executes and checks the VM's basic network functionality.
4. Shut down the VM.
- a) From the Virtual Machine Manager (VMM) interface, select **Virtual Machine**→**Shut Down**→**Shut Down**.
 - b) Close the VM window.
 - c) Close the Virtual Machine Manager window.
-

Topic D

Configure Repositories



EXAM OBJECTIVES COVERED

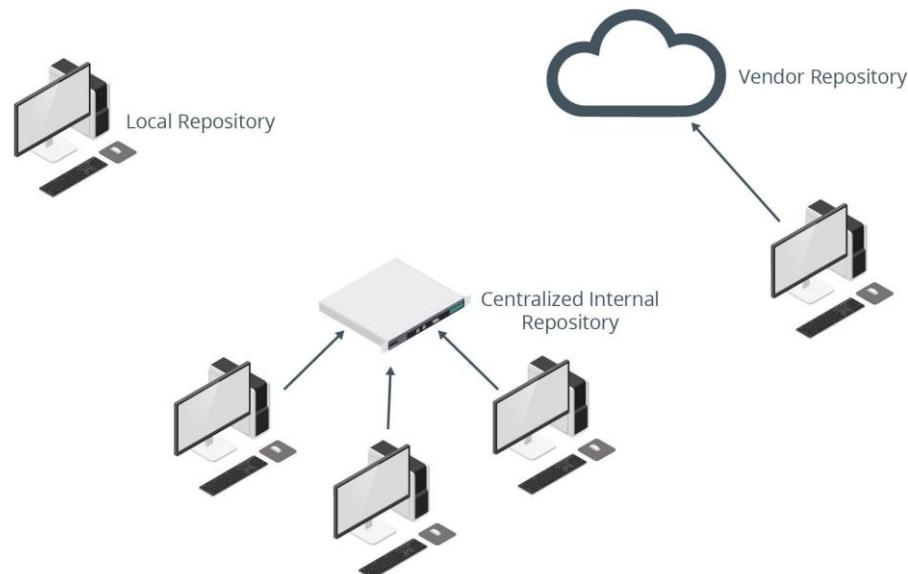
2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

You'll likely acquire most of your software packages from repositories. You may even set up repositories yourself to deliver packages to other systems on your network. In this topic, you'll configure repositories to meet your needs.

REPOSITORIES

Repositories, or repos, are storage locations for available software packages. Repositories can be checked by package managers like YUM and APT when installing or upgrading software. There are three kinds of repositories:

- **Local repositories:** These repositories are stored on the system's local storage drive. Installation is easy, but version control is difficult because of the decentralized nature of local repositories. If you manage 20 Linux servers, each would have its own repository to be maintained.
- **Centralized internal repositories:** These repositories are stored on one or more systems within the internal LAN and managed by the Linux administrator. This centralized approach makes version control much simpler. If you manage 20 Linux servers, one could host the repository and the other 19 could download their packages from it.
- **Vendor repositories:** These repositories are maintained on the Internet, often by the distribution vendor. Version control is very difficult because the vendor decides what package versions are made available.



The types of repositories.

YUM REPOSITORY CONFIGURATION

Administrators can designate a specific location as a YUM repository by using the `createrepo` command. The command updates the XML files that are used to reference the repository location. The repository might be on the local storage drive (a local repository) or available from an Apache web server (centralized internal repository).

After the `createrepo` command is run, a `.repo` configuration file must be created that provides additional information about the repository. The `.repo` files are stored in the `/etc/yum.repos.d/` directory. Some of the components of the `.repo` file are as follows:

`[repo-name]` —The repository name.

`name=Repository Name` —The human-friendly name of the repo. `baseurl=`

—The path to the repo. May be a file (`file:///`) or `http://` path. `enabled=1`

—Enables the repo.

`gpgcheck=0` —Disables GPG checking.

ADDITIONAL YUM SUBCOMMANDS

The `yum` command includes some additional subcommands for viewing and using repositories.

Subcommand	Used To
<code>repolist</code>	See all available repositories.
<code>makecache</code>	Locally cache information about available repositories.
<code>clean all</code>	Clear out-of-date cache information.

REPOSITORY SYNCHRONIZATION

YUM also enables the synchronization of an online repository to a local storage location. This is also known as mirroring. It has the advantage of reducing WAN traffic and lessening the load on the parent repository. The `reposync` utility is used to manage this process. You can choose to synchronize the parent repository once, or cause it to update periodically. For example, an administrator might want to synchronize the Red Hat Network (RHN) repository to a local server, enabling internal Red Hat servers to use the local server as a repository. This local repository might be then configured to update nightly.

SYNTAX

The syntax of the `reposync` command is `reposync [options]`

The following example synchronizes the `server-rpms` repo to a local directory named `packages`:

```
reposync -p packages -r server-rpms
```

APT REPOSITORY CONFIGURATION

Like the YUM package manager, the APT package manager can also be configured to access repositories as part of the software management lifecycle. Repositories are exposed to APT in the `/etc/apt/sources.list` file and in the `/etc/apt/`

sources.list.d/ directory. Like YUM repositories, APT repositories may also be on the local system, on the local network, or hosted on the Internet.

Entries in the **/etc/apt/sources.list** include the following fields, separated by a space:

deb URL distro-name components

Be sure to let APT know about new repositories after editing the **/etc/apt/sources.list** file by running the **apt update** command.



Note: To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

Activity 11-7

Discussing Configuring Repositories

SCENARIO

Answer the following questions to check your understanding of the topic.

1. **What is the purpose of creating a centralized internal repository?**

2. **You have recently taken over support for a group of CentOS Linux servers formerly managed by a different system administrator. You have logged onto each one to check configurations and to update each system. When you attempted to update the systems, no updates were available. Which command can you run to find out which software repositories are configured?**

3. **You have created an internal repository for all your CentOS Linux servers but you are worried that it will be out of date in a few weeks. Downloading a new set of packages every few weeks is a daunting task. Is there any way to automate this process? If so, how?**

4. **You have a few Debian-based Linux systems mixed in with your CentOS ones and you need to configure repositories for those. After adding a new repository to the /etc/apt/sources.list file, which command can you run to let APT know about the new repositories?**

5. **In a YUM repository configuration, what does the gpgcheck=0 entry do?**

Activity 11-8

Configuring Repositories

BEFORE YOU BEGIN

You are logged in to the CentOS GUI as your student account.

SCENARIO

While the Linux vendors tend to provide online repositories, one of the concerns with using these is version control of applications. Develetech has decided to manage an internal repository of software packages, making version control much easier. You will configure a local YUM repository on the CentOS server. You will then make a YUM repository available using Apache HTTP Server.

1. Configure a local YUM repository.
 - a) If necessary, open a terminal.
 - b) Browse to **/Packages** and view the available .rpm files.
 - c) Enter **sudo createrepo /Packages** to designate that directory as a YUM repository.
This may take a few minutes.
 - d) Using **sudo**, create a file named **/etc/yum.repos.d/local-repo.repo** with the text editor of your choice.
 - e) Edit this file by providing the following values:

```
[local-repo]
name=Local Repository
baseurl=file:///Packages
enabled=1
gpgcheck=0
```
 - f) Save and close the file.
2. Verify the location is recognized as a YUM repository.
 - a) Enter **yum clean all**
 - b) Enter **yum repolist** and verify the **local-repo** is displayed.
 - c) Enter **sudo yum -y --enablerepo=local-repo install ksh** to install the KornShell package from the repository.
3. Install Apache HTTP Server to use as a repository.
 - a) Enter **sudo yum -y install httpd**
 - b) Enter **sudo systemctl start httpd** to start the Apache service.
 - c) Enter **systemctl status httpd** and verify that Apache is active (running).
4. Designate Apache as a repository.

- a) Enter `sudo ln -s /Packages /var/www/html/packages` to link the `/Packages` directory to Apache.
- b) Enter `sudo createrepo /var/www/html/packages` to designate the location as a YUM repository.
5. Create the repository reference file.
- a) Using `Sudo`, create a file named `/etc/yum.repos.d/internal-repo.repo` with the text editor of your choice.
- b) Edit this file by providing the following values:
- ```
[internal-repo]
name=Internal Repository
baseurl=http://localhost/packages
enabled=1
gpgcheck=0
```
- c) Save and close the file.
6. Verify the location is recognized as a YUM repository.
- a) Enter `yum clean all`
- b) Enter `yum repolist` and verify the `internal-repo` is displayed.
- c) Enter `sudo setenforce 0`  
This disables SELinux, an access control mechanism that would otherwise prevent access to the web-hosted packages.
- d) Enter `firefox http://localhost/packages` to see a list of packages from the Apache web server.

## Index of /packages

| Name                                                                                                                           | Last modified    | Size | Description |
|--------------------------------------------------------------------------------------------------------------------------------|------------------|------|-------------|
|  <a href="#">Parent Directory</a>           |                  | -    |             |
|  <a href="#">389-ds-base-1.3.7.5..&gt;</a>  | 2018-12-11 18:51 | 1.7M |             |
|  <a href="#">389-ds-base-libs-1.3..&gt;</a> | 2018-12-11 18:51 | 695K |             |
|  <a href="#">ElectricFence-2.2.2..&gt;</a>  | 2018-12-11 18:51 | 35K  |             |
|  <a href="#">GConf2-3.2.6-8.el7.x..&gt;</a> | 2018-12-11 18:51 | 1.0M |             |
|  <a href="#">GeoIP-1.5.0-11.el7.x..&gt;</a> | 2018-12-11 18:51 | 1.1M |             |

- e) Close Firefox when you're done.
- f) Enter `sudo setenforce 1` to re-enable SELinux.
- g) Enter `cd ~` to return to your home directory.

# Topic E

## Acquire Software



### EXAM OBJECTIVES COVERED

*2.1 Given a scenario, conduct software installations, configurations, updates, and removals.*

Repositories are not the only way to download and install software. Several other tools that you'll use in this topic enable you to acquire the files necessary to use software.

### DOWNLOAD SITES

Because of the open source nature of Linux software, it is very common to be able to freely download applications directly from the application vendor. In addition, there are many websites that centralize information about available software, as well as about Linux distributions.

You can search the Internet for Linux software. Here are a few examples of Linux applications that are available for download:

- Audacity®, a music production application.
- Atom, a powerful text editor.
- GIMP, a powerful image editor.
- Nmap, a very useful network mapping utility.

You can also search open source hosting sites like GitHub® for software.

### THE wget AND curl COMMANDS

Most of us are used to accessing websites using a web browser such as Firefox. It is also possible, however, to access websites from the command-line. This is especially useful when downloading a file for which you already know the URL. The **Wget** and **curl** commands can be written into scripts, automating the process of downloading package files.

The following is an example of using **wget** to download a file from the Samba website:

```
wget http://download.samba.org/pub/samba/samba-latest.tar.gz
```

The following is an example of using **curl** to download a file from the Nmap website:

```
curl -o nmap-7.70.tar.bz2 https://nmap.org/dist/nmap-7.70.tar.bz2
```

### DIFFERENCES

While **wget** and **curl** perform the same basic function, there are some key differences:

- **wget** is a command-line utility only, whereas **curl** is implemented using the cross-platform **libcurl** library and is therefore more easily ported to other systems.
- **wget** can download files recursively, whereas **curl** cannot.
- **curl** supports many more network protocols than **wget**, which only supports HTTP/S and FTP.

- **wget** is better suited for straightforward downloading of files from a web server, whereas **curl** is better suited to building and managing more complex requests and responses from web servers.

## SYNTAX

The syntax of the **wget** and **curl** commands is **wget/curl [options] {URL}**

## .tar FILES

Linux often uses two particular utilities to help manage files. The first utility is tape archiver, or **tar**. The second is a compression utility such as **gzip**. The purpose of **tar** is to bundle together multiple files into a single **tarball** with a .tar extension. This makes functions like downloads much easier, since there is only one download necessary to acquire multiple files. The server administrator creates the bundle of files, and whoever downloads the bundle extracts the files from it.

It is essential to know how to work with **tar** because a great deal of the software available for Linux is distributed in tarballs. The following is an example of creating a tarball:

```
tar -cvf tarball.tar file1 file2 file3
```

This bundles **file1**, **file2**, and **file3** into a tarball named **tarball.tar**.

## tar COMMAND OPTIONS

The basic options for the **tar** command are as follows.

| Option    | Used To                                                                                 |
|-----------|-----------------------------------------------------------------------------------------|
| <b>-C</b> | Create the tarball                                                                      |
| <b>-X</b> | Extract the tarball.                                                                    |
| <b>-v</b> | Enable verbose mode.                                                                    |
| <b>-r</b> | Append more files to an existing tarball.                                               |
| <b>-t</b> | Test the tarball or see what files are included in the tarball.                         |
| <b>-f</b> | Specify the name of the tarball in the next argument (must be used as the last option). |

## COMPRESSED FILES

File compression takes one or more files and reduces their size, making downloads far more efficient. There are actually several compression utilities available—**gzip** being one of the common ones in the Linux world. Files compressed with **gzip** take on the .gz extension. It is common to compress a tarball to create the .tar.gz or .tgz extension, which many repositories use when distributing software packages. Another extension you may see is .tar.bz2, indicating that the tarball was compressed with the **bzip2** utility.

The basic commands for **gzip** are:

- **gzip {file name}** —Compresses the file and appends the .gz extension.
- **gzip -d {file name}** —Decompresses the file.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 11-9

## Discussing Acquiring Software

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. **You need to transfer an entire directory, /2018, containing hundreds of files to five other servers. How can you bundle the files into a single file for transfer?**
  
2. **You have a tarball of a program that is 300 MB in size and that is too large for most email programs to handle. You need to somehow make the group of files smaller, but you also don't want to create multiple files to send. What is the solution?**
  
3. **If you downloaded the file wordsmith-12.3.tar.gz, how would you retrieve its contents?**
  
4. **How do you use the wget command to download software?**
  
5. **True or false? You can use the wget command to upload files to a web directory.**

# Activity 11-10

## Acquiring Software

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

### SCENARIO

You are investigating ways of downloading software from the web. Specifically, you are considering writing a script to automate the download process. You will use `Wget` and `curl` to try the downloads manually.

1. Use the `wget` utility to download a file from the web.
  - a) At a terminal, ensure you're in your home directory.
  - b) Enter `wget https://download.samba.org/pub/samba/samba-latest.tar.gz` to download the most recent source code file for the Samba service.
  - c) Check your home directory for a file named `samba-latest.tar.gz`
2. Use `curl` to download a file from the web.
  - a) Enter `curl -o nmap-7.70.tar.bz2 https://nmap.org/dist/nmap-7.70.tar.bz2` to download version 7.70 of the Nmap utility.
  - b) Check your home directory for a file named `nmap-7.70.tar.bz2`
3. Expand a source code tarball so that it is ready to be compiled in a later activity.
  - a) Enter `tar -xvf nmap-7.70.tar.bz2` to extract the files.
  - b) Verify that the source code files were extracted in the `~/nmap-7.70/` directory.

**Note:** You will compile the Nmap utility source files in a later activity. Your current objective is just to acquire and unpack it.

# Topic F

## Build Software from Source Code



### EXAM OBJECTIVES COVERED

2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

Package managers are efficient and convenient forms of acquiring and installing software. However, there may be times when you want to build software from source code—after all, most Linux software is open source.

### WHY COMPILE?

Software that is packaged as an .rpm or a .deb is pre-compiled by the vendor. Usually, this packaged software is configured with generic settings and options. A Linux user may want to modify the software, for example, to optimize it for their specific hardware. This may result in maximum performance. Packaged software is most common in Linux, but compiling your own software is a normal Linux task. The same is not true for Windows and macOS, where nearly all software is pre-compiled (because you don't typically have access to the open source code).

### COMPILERS

Compiling software manually does not use a management tool as packaged software does. To compile software, there must be a compiler installed. Compilers translate source code written in a human-friendly programming language, such as C or C++, into machine-readable binaries.

A common compiler for Linux is the GNU Compiler Collection (GCC), implemented as the `gcc` utility. There are often other supporting libraries that must also be in place. The required libraries vary by the software being compiled. Typically, software developers will provide a list of these necessary libraries with the application source code. These may be header files (.h file extension) or library files (.a file extension).

### LIBRARIES

Program **libraries** are chunks of compiled code that can be used in programs to accomplish specific common tasks. Shared libraries enable more modular program builds and reduce time when compiling the software. Compiled software must be able to access needed libraries when it runs.

Shared libraries are typically included with a Linux distribution and are placed in the `/usr/lib/` directory for general accessibility. Libraries that need to be accessed by essential binaries are typically placed in the `/lib/` directory. Libraries that aren't packaged with a distro can also be included by the developer to ensure that the user can run their program.

### THE LDd COMMAND

The `ldd` command enables a user to view shared library dependencies for an application. This can be useful for troubleshooting or gathering information about system requirements for an application.

```

Shared libraries
[root@server01 ~]# ldd /usr/local/bin/nmap
 linux-vdso.so.1 => (0x00007ffdbe5eb000)
 libdl.so.2 => /lib64/libdl.so.2 (0x00007f4f813dc000)
 libstdc++.so.6 => /lib64/libstdc++.so.6 (0x00007f4f810d5000)
 libm.so.6 => /lib64/libm.so.6 (0x00007f4f80dd3000)
 libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x00007f4f80bbd000)
 libc.so.6 => /lib64/libc.so.6 (0x00007f4f807f0000)
 /lib64/ld-linux-x86-64.so.2 (0x00007f4f815e0000)
[root@server01 ~]#

```

*Viewing a program's shared libraries.*

## SYNTAX

The syntax of the **l****d****d** command is **l****d****d** [options] {program binary}

## THE SOFTWARE COMPILED PROCESS

When an administrator downloads software, there is a common process to build the executable file called **software compilation**:

1. Unpack the download, typically using **tar** and/or **gzip** commands.
2. Change into the directory that gets created as part of the unpacking process.
3. Run the **./configure** command to gather system information needed by the application. This information is stored in the **makefile**, which is referenced during the next step.
4. Use the **make** command to compile the application using the information stored in the makefile. Note that this usually requires root privileges.
5. Use the **make install** command to install the resulting binaries (the application).

Many developers will provide instructions and options that may modify this process somewhat. Specifically, there may be options or modifications that can be made to the makefile before the **make** command is run to optimize the software for the system or the user's needs. It is important to review any README or other instruction files that are downloaded as part of the software.

## THE **make** COMMAND

In most cases, once the makefile is created, simply issuing **make** and then **make file** without arguments will install the application. This is because the **make** command automatically looks for the makefile in the current directory. You can, however, issue **make** with various options.

## MORE ON MAKEFILES

A makefile is a file that contains instructions used by a compiler to build a program from source code. These instructions typically define the resources that the program depends on in order to function properly, as well as any additional directives as

defined by the developer. In the following simple example, the program executable **myprog** depends on two object files, **mymain.o** and **myfunc.o**:

```
myprog: mymain.o myfunc.o
 gcc -o myprog mymain.o myfunc.o
```

```
mymain.o: mymain.c
 gcc -c mymain.c
```

```
myfunc.o: myfunc.c
 gcc -c myfunc.c
```

On the second line, **gcc** compiles the objects necessary for the program to run. On the remaining lines, each object is associated with a C source code file, then compiled using that source file. Using this approach, if you make changes to a single C source file (e.g., **mymain.c**), the **make** command will be able to efficiently rebuild the program based on the directives in the **makefile**.

 **Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 11-11

## Discussing Building Software from Source Code

### SCENARIO

Answer the following questions to check your understanding of the topic.

---

1. **What does the `./configure` command do?**
  
  2. **You have downloaded, decompressed, and extracted a software program tarball, and run the `./configure` command. What is the next step in the installation process?**
  
  3. **Which command actually compiles the source code into executable code?**
  
  4. **What is the final step in compiling software?**
  
  5. **What does the `make install` command do in the software compilation process?**
-

# Activity 11-12

## Compiling and Installing an Application

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. Previously, you downloaded source files for Nmap.

### SCENARIO

Develetech will be relying on Nmap to troubleshoot its networked systems and perform vulnerability assessments. You know that compiling Nmap from source code enables greater flexibility and control. You will do a basic software compile of Nmap.

1. Confirm that Nmap is not currently installed.
  - a) Enter **rpm -qi nmap**
  - b) Verify that **nmap** is not installed.
2. Install the necessary GCC compiler for Nmap on CentOS 7.
  - a) Enter the following:  
**sudo yum -y install gcc-c++ --disablerepo=internal-repo**
  - b) Wait for the package to finish installing.
3. Compile the Nmap source code.
  - a) Change to the **~/nmap-7.70** directory.
  - b) Enter **./configure** to generate a makefile based on your system's configuration.  
This may take a few minutes.
  - c) Enter **make** to compile the software based on the makefile instructions.
  - d) Enter **Sudo make install** to install the binaries on the system.
  - e) Enter **/usr/local/bin/nmap** and verify that it is installed.

# Topic G

## Troubleshoot Software Dependency Issues



### EXAM OBJECTIVES COVERED

4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

One of the biggest issues when it comes to managing software is dependencies. You need to ensure that all dependencies are accounted for in order for applications to work as intended. So, you'll identify some common dependency issues and implement solutions.

### DEPENDENCY TROUBLESHOOTING

Dependency troubleshooting involves discovering what package dependencies exist before attempting a deployment and ensuring that the needed dependencies are stored in the repositories. Troubleshooting repository issues usually starts by verifying network access to the repositories, and then checking the repository configuration files. On a Red Hat-derivative distribution, for example, these files will be located in the `/etc/yum.repos.d/` directory.

When troubleshooting problems while compiling software, begin by verifying that the appropriate compiler, compiler version, and supporting libraries are present.

### HOW YUM AND APT MANAGE DEPENDENCIES

The YUM and APT package managers rely on information inside the packages to discover what dependencies the software might have. YUM and APT then search the repository for the additional packages that are needed. Both can be configured to automatically install the dependencies. This ability greatly simplifies the installation process. YUM and APT essentially interpret the installation commands as "install this software and anything else it might need."

### GUIDELINES FOR TROUBLESHOOTING SOFTWARE DEPENDENCY ISSUES

Use the following guidelines when troubleshooting software dependency issues.

### TROUBLESHOOT SOFTWARE DEPENDENCIES AND REPOSITORIES

When troubleshooting software dependencies and repositories:

- Use the `rpm -V {package name}` command to verify that all components of a package are installed. This is particularly useful if you believe configuration files or other needed files have been deleted.

- Use the `rpm -qR {package name}` and `yum deplist {package name}` commands to discover dependencies before attempting an installation on Red Hat-derivative distributions.
- Use the `apt-cache depends {package name}` command to discover dependencies before attempting an installation on Debian-derivative distributions.
- Use repositories and ensure all dependency packages are stored in the repository along with the primary software package.
- Ensure your repositories are current.
- Ensure systems have network connectivity to repositories.

## TROUBLESHOOT PATCHING AND UPDATE ISSUES

When troubleshooting patching and update issues:

- Read patch documentation before applying the patch. You must use this information to plan for potential disruption (e.g., downtime as a result of a required restart).
- Ensure there is network connectivity when retrieving updates.
- Test patches and updates before deploying them to production systems to ensure they work as expected.
- Ensure dependencies and software versions are satisfied for the patches and updates.
- Check installation logs if systems experience issues after patching.
- Have a contingency plan in case patches result in unexpected behavior, even if that behavior is not immediately noticeable. Ensure you can roll back to a pre-patched version and that any stored data that could be affected by a patch issue is backed up beforehand.

## TROUBLESHOOT GCC AND LIBRARY ISSUES

When troubleshooting GCC and library issues:

- When compiling software, check the documentation for required GCC or other compiler versions.
- Verify any required versions of the Linux kernel and software dependencies that the compiled program has.
- Use `ldd [options] {program binary}` to check for shared library file dependencies for software you will be compiling.
- Verify library file versions and availability.
- Consider compiling and testing a program in a virtual machine to ensure it runs properly.
- Assume root privileges when using the `make install` command.

# Activity 11-13

## Discussing Software Dependency Issues

# SCENARIO

Answer the following questions to check your understanding of the topic.

1. **What is a software dependency?**
  2. **You have a CentOS system and you need to install an application that you know has a lot of dependencies. What can you do to satisfy those dependencies and install the application?**
  3. **You have a CentOS system that runs an application for your company, but anyone who runs it receives errors, as if some critical parts of the application were missing. How can you verify the application's components?**
  4. **Being security conscious, you always check out any software package being installed to your CentOS systems. Which command can you run to check supporting software packages prior to installing a new software application?**
  5. **Why do you need to assume root privileges when running the make install command?**

# Activity 11-14

## Troubleshooting Dependency Issues

### SCENARIO

Develetech wants to ensure that the `yum` and `apt` installation tools are effectively used and understood. Answer the following questions to help clarify the proper use of the tools.

1. **In what way are yum and apt more effective at managing software dependencies than rpm or dpkg?**
  
2. **Michael Anderson, a Develetech employee, calls you for help with software. He believes some files may have been deleted from his CentOS system, and now a piece of software he needs does not run. What command could you run to check that all the necessary components of the software are installed on the system?**
  
3. **Andrew Riley, a Develetech Marketing department employee, has been given privileges for installing software on his Ubuntu laptop. He has downloaded a package and is using the `rpm -ivh <package name.rpm>` command to install the software. He says the system error indicates that the `rpm` command is not found. What is the issue?**

## Summary

In this lesson, you managed packages using package managers and explored various options for creating and maintaining the repositories that provide these packages. This will enable you to easily install additional software on Linux systems.

**What packages might you need to install in your organization? Which package management tools will you use most often?**

**As an administrator, why might you create your own repositories?**



**Practice Question:** Additional practice questions are available on the course website.

# Lesson 12

## Securing Linux Systems

**LESSON TIME: 4 HOURS**

### LESSON INTRODUCTION

The importance of cybersecurity cannot be overstated. This is true of all organizations, no matter their size or what industry they're in. After all, numerous information breaches and other attacks have dominated the headlines in recent years. Because Linux® systems store and process much of the world's data, including data of a sensitive nature, those Linux systems need to be secured. In this lesson, you'll use multiple techniques to bolster the security of your Linux systems and therefore minimize the risk of the organization falling victim to an attack.

### LESSON OBJECTIVES

In this lesson, you will:

- Implement best practices for cybersecurity in Linux.
- Implement identity and access management (IAM) methods in Linux.
- Configure Security-Enhanced Linux (SELinux) or AppArmor.
- Configure firewalls to filter network traffic.
- Implement event and system logging services.
- Back up, restore, and verify data.

# Topic A

## Implement Cybersecurity Best Practices



### EXAM OBJECTIVES COVERED

3.3 Summarize security best practices in a Linux environment.

There are some foundational best practices for securing systems that apply to Linux, as well as operating systems in general. You'll start your security tasks by identifying and implementing those best practices.

### CYBERSECURITY

**Cybersecurity** refers to the protection of computer systems and digital information resources from unauthorized access, attack, theft, or data damage. As a business discipline, cybersecurity is pivotal for essentially all modern organizations, no matter their size. It is also an essential practice for individuals whose identities are inseparable from the digital space, especially in the highly connected ecosystem that is the Internet.

As with any computer system, Linux systems must be subjected to sound cybersecurity practices in order for them to function with minimal risk to the organization. While your organization may employ security specialists, anyone who touches a computer system, especially those with administrative access to sensitive assets, is responsible for security.

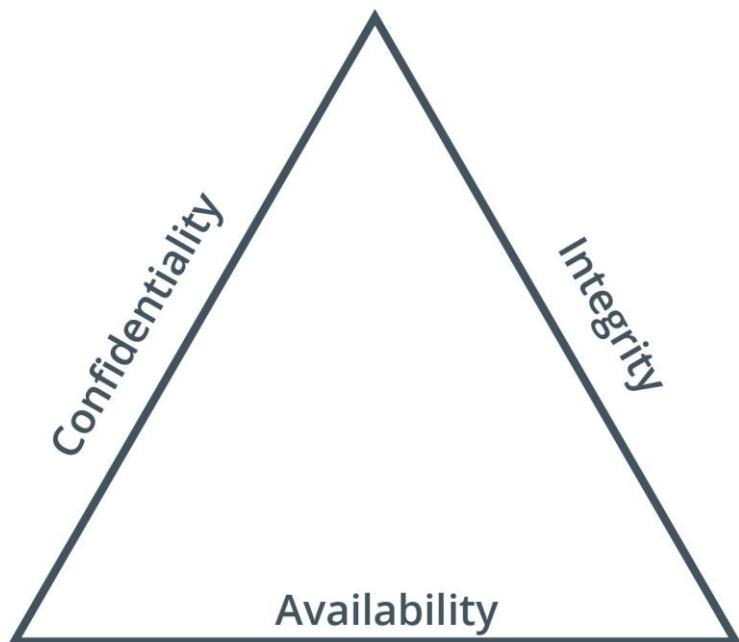
### THE CIA TRIAD

Cybersecurity seeks to address three specific principles: confidentiality, integrity, and availability. This is called the **CIA triad**. If one of the principles is compromised, the security of the organization is threatened.

The CIA triad consists of three principles.

| Principle              | Description                                                                                                                                                                                                                                                                                                                                      |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Confidentiality</b> | This is the fundamental principle of keeping information and communications private and protected from unauthorized access.<br><br>Confidential information includes trade secrets, personnel records, health records, tax records, and military secrets.<br><br>Confidentiality is typically controlled through encryption and access controls. |

| Principle           | Description                                                                                                                                                                                                                                                                               |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Integrity</b>    | <p>This is the fundamental principle of keeping organizational information accurate, free of errors, and without unauthorized modifications.</p>                                                                                                                                          |
|                     | <p>For example, if an attack on a school system's server occurred and student test scores were modified, the integrity of the grade information would be compromised by unauthorized modification.</p>                                                                                    |
|                     | <p>Integrity is typically controlled through hashing, digital signatures, certificates, and change control.</p>                                                                                                                                                                           |
| <b>Availability</b> | <p>This is the fundamental principle of ensuring that computer systems operate continuously and that authorized persons can access the data that they need.</p>                                                                                                                           |
|                     | <p>Information available on a computer system is useless unless the users can get to it. Consider what would happen if the Federal Aviation Administration's air traffic control system failed. Radar images would be captured but not distributed to those who need the information.</p> |
|                     | <p>Availability is typically controlled through redundancy, fault tolerance, and patching.</p>                                                                                                                                                                                            |



The CIA triad.

## AUTHENTICATION METHODS

**Authentication** is the verification of an individual's identity. It is a prominent component in cybersecurity because it enables an organization to trust that users are who they claim to be. There are various ways to authenticate a user, some of which you should be familiar with.

| Authentication                             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PINs, passwords, and passphrases           | These are all strings of text that you input along with a user name in order to sign in to a system. Depending on the system, you may be constrained by the type of characters you can use, the minimum or maximum number of characters you can use, and more. In most cases, each user memorizes their own password and is the only one to know it.                                                                                                                                                                                                                                                                               |
| <b>Tokens and one-time passwords (OTP)</b> | A token is any unique object, whether physical or digital, that you possess and that can be used to verify your identity. Tokens are typically used to generate one-time passwords (OTP), which are passwords that either expire after first use or expire within a small time period, or both. In either case, OTPs are not meant to be memorized like normal passwords. Tokens can also leverage digital certificates as the authentication information.                                                                                                                                                                         |
| <b>Biometrics</b>                          | A hardware token is a physical device that generates and stores the authentication information, and that information is tied to that particular device. One common example is a key fob that generates and displays a numeric token on the key fob's small screen. RSA SecurID® is the most popular security key fob.                                                                                                                                                                                                                                                                                                              |
| <b>RADIUS and TACACS+</b>                  | Software tokens, on the other hand, are generated by a system that can distribute the authentication information to any authorized general-purpose device —like a smartphone or a desktop computer. RSA SecurID also has a mobile app for this purpose, though apps like Google Authenticator™ can also generate one-time tokens.                                                                                                                                                                                                                                                                                                  |
|                                            | These are authentication schemes that verify a user's identity based on their physical characteristics. This can involve fingerprint scanners, iris scanners, hand geometry scanners, voice recognition and facial recognition software, and many more. They tend to be much harder to compromise than passwords and tokens, as a person's exact physical characteristics are difficult to replicate.                                                                                                                                                                                                                              |
|                                            | Remote Authentication Dial-In User Service (RADIUS) is an Internet standard protocol that provides authentication, authorization, and accounting (AAA) services. When a network contains several remote access servers, you can configure one of the servers to be a RADIUS server, and all of the other servers as RADIUS clients. The RADIUS clients will pass all authentication requests to the RADIUS server for verification.<br><br>Terminal Access Controller Access-Control System (TACACS) and its successor, TACACS+, also provide AAA services for remote users. TACACS+ is more secure and more scalable than RADIUS. |

| Authentication  | Description                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LDAP</b>     | <p>Lightweight Directory Access Protocol (LDAP) is a directory service protocol that runs over Transmission Control Protocol/Internet Protocol (TCP/IP) networks. LDAP clients authenticate to the LDAP service, and the service's schema defines the tasks that clients can and cannot perform while accessing a directory database, the form the directory query must take, and how the directory server will respond.</p>   |
|                 | <p>Secure LDAP (LDAPS) is a method of implementing LDAP using Secure Sockets Layer/Transport Layer Security (SSL/TLS) encryption protocols to protect the confidentiality and integrity of directory service transmissions.</p>                                                                                                                                                                                                |
| <b>Kerberos</b> | <p>This is an authentication service that is based on a time-sensitive ticket-granting system. It is used as a single sign-on (SSO) method where the user enters access credentials that are then passed to the authentication server, which contains an access list and allowed access credentials. Kerberos can be used to manage access control to many different services using one centralized authentication server.</p> |

## LINUX KERBEROS COMMANDS

The Linux implementation of Kerberos has a few commands of note:

- **kinit**—Authenticates with Kerberos, granting the user a ticket granting ticket (TGT) if successful.
- **kpassword**—Changes the user's Kerberos password.
- **klist**—Lists the user's ticket cache.
- **kdestroy**—Clears the user's ticket cache.

For example, issuing **kinit user@domain.tld** will prompt for the user's password that is stored in the directory server database. If the correct password is provided, then the user will obtain a ticket from the Kerberos server. The user can then issue **klist -v** to verify that the ticket was obtained. The following is a sample part of the output:

Credentials cache: API:501:9

Principal: user@domain.tld

Cache version: 0

Server: krbtgt/domain.tld@domain.tld

Client: user@domain.tld

Ticket etype: aes128-cts-hmac-sha1-96

Ticket length: 256

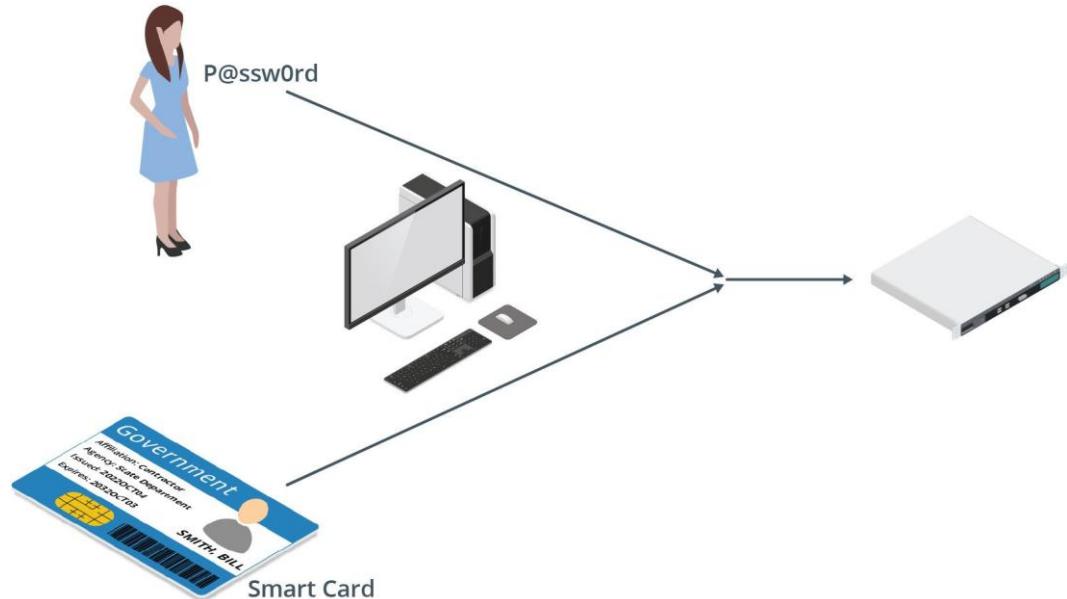
## MULTI-FACTOR AUTHENTICATION

Authentication methods can make use of several factors. These factors are typically expressed as something you know, something you have, and something you are.

**Multi-factor authentication (MFA)** is the practice of requiring the user to present at least two different factors before the system authenticates them. This helps prevent unauthorized access should one factor be compromised, like an attacker guessing a user's password. Tokens and OTPs (something you have) are commonly used as the

second factor after the user's standard password. On more advanced systems, biometrics (something you are) are also used as a factor.

In order for a system to be MFA, it must incorporate more than one *factor*, not more than one method. For example, using a hardware token and a software token would not qualify, because they are the same factor (something you have).



*Using a password and a smart card as two different factors for logging in to a system.*

## PRIVILEGE ESCALATION

**Privilege escalation** occurs when a user is able to obtain access to additional resources or functionality that they are normally not allowed access to. One of the most common scenarios is when a normal user is able to exploit some vulnerability on a system to gain root-level privileges.

Although privilege escalation can be used for legitimate purposes, e.g., an administrator assuming root privileges through `sudo`, you must be on the lookout for any behavior that enables attackers to escalate their privileges. One pitfall that can enable such behavior is poorly configured SUID and SGID permissions.

While changing the permissions of a file to use either SUID or SGID, consider the following:

- Use the lowest permissions needed to accomplish a task; i.e., adhere to the principle of least privilege. It is recommended not to give a file the same SUID or SGID as the root user. A user with fewer privileges is often enough to perform the task.
- Watch for back doors. If the user runs a program with the SUID set to root, then the user retains root as the effective user ID when the user goes through the back door. For example, some programs enable an attacker to shell out to a remote system.

## chroot JAIL

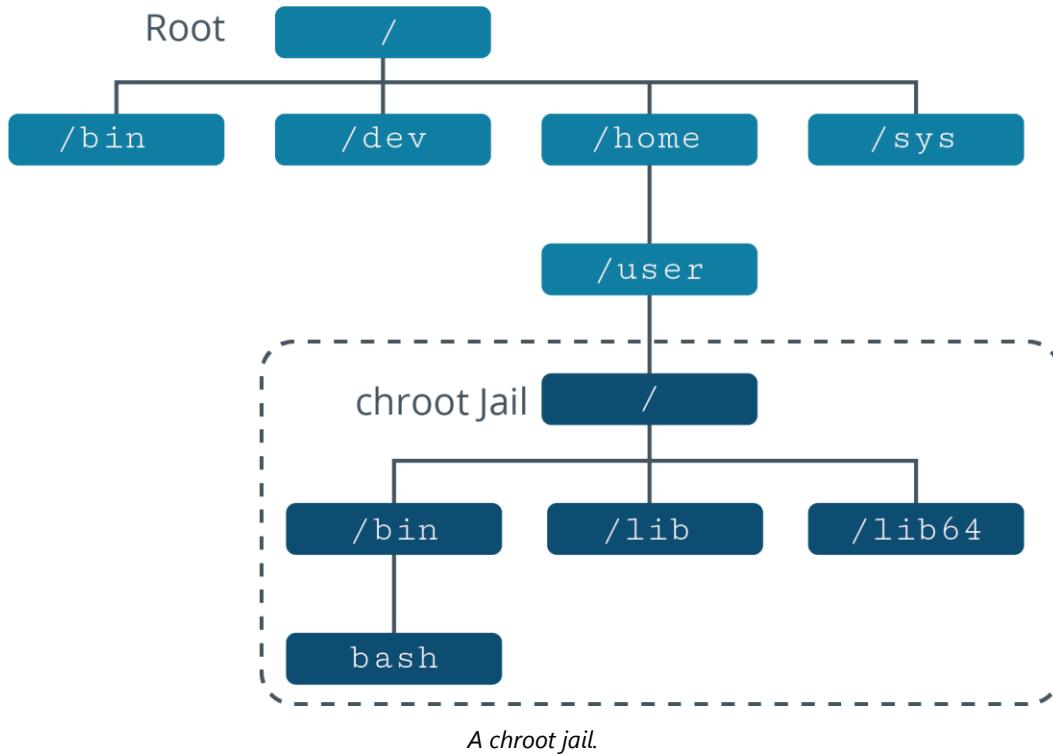
A **chroot jail** is a technique of controlling what a process—a user, for example—can access on a file system by changing the root directory of that process's environment. This new root directory is called a "jail" because the process and any child processes that it spawns will be unable to "break out" of that location and access other parts of

the file system. For example, if you change a process's root location to `/home/ user/` then, when it references the root (`/`), the process will be confined to `/home/ user/` instead of the actual root of the file system. This is useful in separating privileged access on the file system so that a malicious or rogue process cannot cause damage outside of its jail.

The **chroot** command is used to actually change the root directory for an environment. For example, **chroot /home/user /usr/bin/bash** will create the new root directory using the Bash shell as the process inside the jail.



**Note:** A chroot jail will not prevent a user or process with root privileges from breaking out of that jail.



## SYNTAX

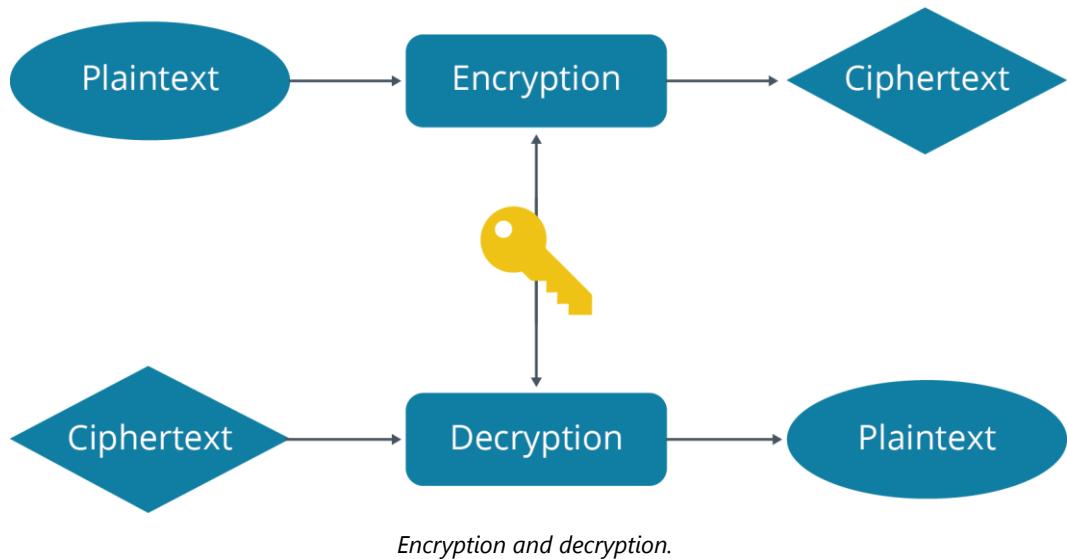
### X

The syntax of the **chroot** command is **chroot [options] {new root directory} [command]**

## ENCRYPTION

**Encryption** is a cryptographic technique that converts data from plaintext form into coded, or ciphertext, form. Decryption is the companion technique that converts ciphertext back to plaintext. An algorithm called a cipher is responsible for the conversion process.

When a message is encrypted, only authorized parties with the necessary decryption information can decode and read the data. This information is called a key, and it is used with the cipher to ensure the message is unreadable to those not in possession of the key. Encryption is therefore one of the most fundamental cybersecurity techniques for upholding the confidentiality of data.



## **TYPES OF ENCRYPTION**

Encryption can be applied to data in transit (passing through a network), data in use (accessed in memory), and data at rest (stored on a device). There are several subtypes of data at rest encryption, with two of the most prominent being:

- Full drive/disk encryption (FDE), which encrypts an entire storage drive, partition, or volume using either hardware or software utilities.
- File encryption, which encrypts individual files and folders on a file system using software utilities.

## **LUKS**

**Linux Unified Key Setup (LUKS)** is a platform-independent FDE solution that is commonly used to encrypt storage devices in a Linux environment. On Linux, LUKS uses the dm-crypt subsystem that was incorporated in the Linux kernel around version 2.6. This subsystem creates a mapping between an encrypted device and a virtual device name that user space software can work with. LUKS offers a high degree of compatibility with various software because it standardizes the format of encrypted devices.

## **THE shred COMMAND**

Before encrypting a device, it's a good idea to overwrite its contents with random data or all zeros. This ensures that no sensitive data from past use remains on the device. The **shred** command can be used to securely wipe a storage device in this manner.

## **THE cryptsetup COMMAND**

The **cryptsetup** command is used as the front-end to LUKS and dm-crypt. The LUKS extensions to **cryptsetup** support various actions, including the following.

| <b>LUKS Action</b> | <b>Used To</b>                                                                                        |
|--------------------|-------------------------------------------------------------------------------------------------------|
| luksFormat         | Format a storage device using the LUKS encryption standard.                                           |
| isLuks             | Identify if a given device is a LUKS device.                                                          |
| luksOpen           | Open a LUKS storage device and set it up for mapping, assuming the provided key material is accurate. |

| LUKS Action | Used To                                        |
|-------------|------------------------------------------------|
| luksClose   | Remove a LUKS storage device from mapping.     |
| luksAddKey  | Associate new key material with a LUKS device. |
| luksDelKey  | Remove key material from a LUKS device.        |



```
root@server01 ~]# cryptsetup -v luksFormat /dev/sda6
WARNING!
=====
This will overwrite data on /dev/sda6 irrevocably. Storage device to encrypt

Are you sure? (Type uppercase yes): YES
Enter passphrase:
Verify passphrase:
Command successful.
[root@server01 ~]#
```

*Encrypting a storage device.*

## SYNTA

### X

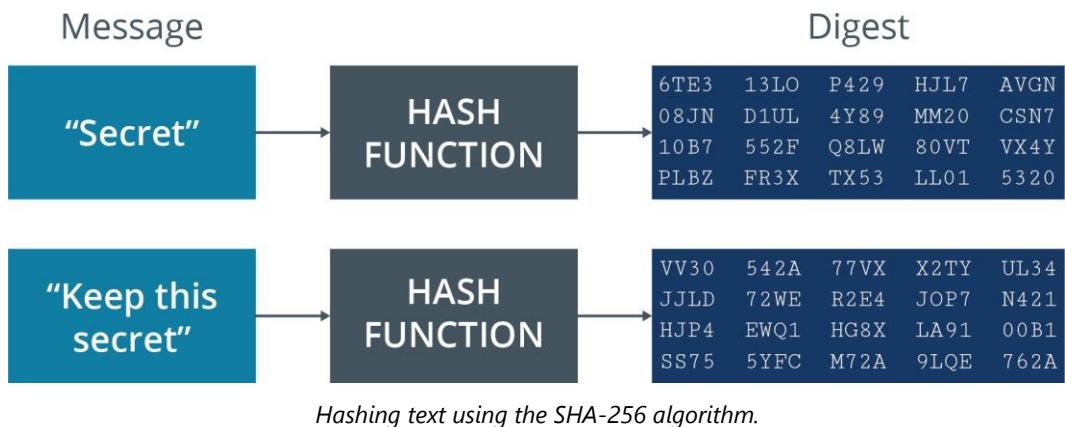
The syntax of the `cryptsetup` command is `cryptsetup [options] {action} [action arguments]`

## HASHING

**Hashing** is a process or function that transforms plaintext input into an indecipherable fixed-length output and ensures that this process cannot be feasibly reversed. The resulting output of the hashing process is called a **hash**, **hash value**, or **message digest**. The input data can vary in length, whereas the hash length is fixed. In a hash function, even the smallest of changes to data can modify the hash considerably, making it much more difficult to deduce the cryptographic material based on certain patterns. When comparing a value against its hash (to verify the value hasn't been changed), if the hash you generate matches the hash provided with the value, you can be pretty sure the value was not modified.

Hashing has several uses:

- It is used in a number of password authentication schemes.
- A hash value can be embedded in an electronic message to support data integrity.
- A hash of a file can be used to verify the integrity of that file after transfer.



## NETWORKING SECURITY BEST PRACTICES

The following describes some best practices you should incorporate in your networking configurations:

- Enable SSL/TLS in all web server technology. This guarantees confidentiality and authenticity in the data that is sent to and received from clients. This is *especially* important in websites that deal with sensitive data, like credit card information, personally identifiable information, etc.
- Configure SSH to disable root access. This can prevent an authorized user from gaining complete access over a system from a remote location. Instead of enabling root access, assign sudoer privileges to the necessary accounts.
- For remote access and receiving other types of network connections from clients, configure the system to, by default, deny hosts that it does not recognize. You can create a whitelist of acceptable hosts so that all hosts not on the list are automatically untrusted.
- Consider changing the default port associations for certain services, like SSH and HTTP/S. This might be able to confound attackers or temporarily stop automated attacks that target well-known ports; however, it's important to note that actions like this—called security through obscurity—are not effective when done in isolation. In some cases, changing default ports may be more hassle than it's worth.

## ENABLING SSL/TLS IN APACHE

The following is a general process for enabling SSL/TLS for use with the Apache web service:

1. Generate a self-signed certificate using a tool like OpenSSL, or request and obtain a certificate from an external authority.
2. Download and install the `mod_ssl` package.
3. Open the `/etc/httpd/conf.d/ssl.conf` file for editing.
4. Find the `<VirtualHost _default_:443>` line and uncomment the `DocumentRoot` and `ServerName` lines, then replace their values as necessary.
5. Below this, ensure `SSLEngine` is set to `On`.
6. Point `SSLCertificateFile` to the path where your certificate file is located.
7. Point `SSLCertificateKeyFile` to the path where your private key file is located.
8. Restart Apache.
9. Open a browser and verify that the site is presenting a certificate.

## USER ACCESS SECURITY BEST PRACTICES

The following describes some best practices you should incorporate when managing user access:

- Protect the boot loader configuration with a password to prevent unauthorized personnel from tampering with boot options.
- Enable a password within your system's BIOS/UEFI to prevent unauthorized personnel from installing and/or booting into a new operating system.
- Consider discouraging the use of USB devices, particularly USB storage devices like thumb drives. USB thumb drives can make it easy for an insider threat to exfiltrate sensitive data from a system, or to load malware onto that system. You can also explicitly block USB access by unloading the relevant modules from the kernel. Use `lsmod` to search for `usb_storage` and any dependent modules. Then use `modprobe -r <module name>` to unload the relevant modules from the kernel. You can also prevent the relevant modules from being loaded at boot by creating a blacklist file in `/etc/modprobe.d/` that contains the line:

```
install <module name> /bin/false
```

- Ensure that user IDs (UIDs) are not being shared and are instead unique to each user. By sharing UIDs, your ability to audit user actions is compromised, as you cannot maintain accountability for each individual user.
- Consider establishing a public key infrastructure (PKI) that can enforce the use of private and public keys for authentication. This creates a password-less login scheme to mitigate password cracking techniques used by attackers to gain access to an account.
- Restrict access to cron, the Linux job scheduler. This can prevent unauthorized users from configuring the system to automatically run a malicious or unwanted task every so often, bypassing the need to log in and manually issue a command. You can add user names to the `/etc/cron.d/cron.deny` file to blacklist these users from accessing cron. Each user name must appear on its own line. To whitelist users, create the `/etc/cron.d/cron.allow` file and add the names of authorized users. All other users will be prevented from accessing cron.
- Disable the use of **Ctrl+Alt+Del** to prevent users from rebooting a system and disrupting service availability. On systemd systems, you can mask `ctrl-alt-del.target` to disable **Ctrl+Alt+Del** functionality: `systemctl mask ctrl-alt-del.target`

## ADDITIONAL SECURITY BEST PRACTICES

The following describes some additional best practices you should consider implementing in your Linux systems.

- Enable the `auditd` service to ensure that records used in auditing are being written to storage. These records include everything from number of failed logins, number of commands issued, and much more. Use the `aureport` and `ausearch` commands to see auditing data. Enter `systemctl enable auditd` to enable `auditd` at boot.
- Add a banner message to `/etc/issue` that will display useful information every time a user logs in. This information can include what purpose the system serves, any behavior the administrator expects the user to adhere to, etc. You can also edit the message of the day (MOTD) in `/etc/motd` to display more information below the banner. To display a message to SSH clients, edit the `/etc/issue.net` file.
- Separate operating system data and other types of data, like application files, into different partitions. Segmenting data can help increase the availability of one type

of data should another type be inaccessible; for example, if a partition containing application data were to become corrupted, the system partition can still continue to function. A common approach is to create separate partitions for the root file system (`/`) and the `/home` directory. User-specific data is therefore segmented from the rest of the file system.

- Regularly monitor the Common Vulnerabilities and Exposures (CVE) database, a public dictionary of vulnerabilities that facilitates the sharing of data among organizations, security tools, and services. CVE monitoring enables you to stay current on the latest vulnerability trends that might affect your Linux systems. The official CVE website from the MITRE Corporation (<https://cve.mitre.org/>) and the National Vulnerability Database (NVD) (<https://nvd.nist.gov/>) are both useful sources for identifying CVE entries.
- Harden your system by disabling or uninstalling unused and/or insecure services. Only enable services that the system needs, and try to only use services that use security techniques like cryptography and that are patched. Services that are popular vectors for attack can include, but are not limited to:
  - FTP—Default FTP configurations do not use encryption and therefore send data in cleartext over the network. They can also be used for data exfiltration and the spreading of malware. Most modern systems use a more secure FTP daemon like `vsftpd`, but if your system is using standard FTP, you should use a package manager like YUM to uninstall the `ftp` package.
  - Telnet—Similar to FTP, remote connections using Telnet do not encrypt data and passwords used in authentication. Telnet is not installed on many modern distributions, but if it is, you can uninstall the `telnet` package or set `disabled = yes` in the `/etc/xinetd/telnet` file.
  - Finger—This is an outdated service used to retrieve the status of hosts and users over a network. It is susceptible to many attacks and should not be used. Like standard FTP and Telnet, this is likely not installed on your system. If it is, uninstall the `finger` package or set `disabled = yes` in the `/etc/xinetd/finger` file.
  - Sendmail—This mail transfer agent (MTA) has been susceptible to many vulnerabilities over the years, including buffer overflows and race conditions that could expose sensitive data. This is still included in some modern distros, and can be removed by uninstalling the `sendmail` package or removing its binary manually.
  - Postfix—This is another MTA that you should consider disabling if mail is not needed on the system. If this is included in your system, you can uninstall the `postfix` package or disable its service using `systemctl stop postfix` and then `systemctl disable postfix`

## GUIDELINES FOR IMPLEMENTING CYBERSECURITY BEST PRACTICES

Use the following guidelines when implementing cybersecurity best practices.

### IMPLEMENT CYBERSECURITY BEST PRACTICES

When implementing cybersecurity best practices:

- Protect the CIA of information: confidentiality, integrity, and availability.

- Consider using advanced authentication methods like LDAP and Kerberos to centralize authentication.
- Consider requiring multi-factor authentication for sensitive accounts.
- Ensure you are not granting more access than is necessary when setting SUID and SGID properties.
- Place unprivileged processes in chroot jails to prevent them from accessing other parts of the file system.
- Encrypt sensitive data in transit, in use, and at rest.
- Use LUKS to fully encrypt storage devices.
- Implement networking best practices like limiting root access over SSH.
- Implement user access best practices like ensuring that users don't share UIDs.
- Implement additional best practices like separating OS data from app data on different partitions/volumes.

# Activity 12-1

## Discussing Cybersecurity Best Practices

### SCENARIO

Answer the following questions to check your understanding of the topic.

---

1. **True or false? If one of the CIA triad principles is compromised, the security of the organization is threatened.**
  
  2. **Which authentication method would you use if you didn't want to issue hardware tokens, install special software, or require passwords of your users?**
  
  3. **Why is multi-factor authentication effective in securing network and computing assets?**
  
  4. **What is the purpose of using a chroot jail?**
  
  5. **Why is encryption essential to secure communications on a system or over the network?**
-

# Activity 12-2

## Encrypting a Volume

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have a `/backup/data` volume.

### SCENARIO

The data you'll be backing up to your various logical volumes is sensitive in nature and should not be readable if it were to fall into the wrong hands. To protect the confidentiality of your backed up data, you'll encrypt the volumes that hold this data. You'll start with the `databk` volume. Without the correct key (e.g., a passphrase), a user will only see the scrambled ciphertext of this volume, and will be unable to read the plaintext data of individual files.

1. Prepare the data backup volume for encryption.
  - a) Enter `sudo umount /backup/data`
  - b) Enter `sudo shred -v --iterations=1 /dev/backup/databk`

This will overwrite the contents of the volume to securely wipe any existing data. This is a good practice to ensure that no sensitive data remains before you prepare the encrypted volume.

- c) Verify that the `shred` command finishes successfully.
2. Encrypt the data backup volume with a passphrase.
  - a) Enter the following:  
`sudo cryptsetup -v --verify-passphrase luksFormat /dev/backup/databk`
  - b) Enter **YES** when prompted to confirm.
  - c) When prompted for a passphrase, enter `linuxplus`
  - d) Verify the passphrase.
  - e) Verify that the command was successful.
3. Open the encrypted volume and verify that it is listed.
  - a) Enter the following:  
`sudo cryptsetup luksOpen /dev/backup/databk databk`
  - b) Enter `linuxplus` as the passphrase.
  - c) Verify that you are returned to a prompt without errors.
  - d) Enter `ls -l /dev/mapper | grep databk`
  - e) Verify that the encrypted volume is listed.
4. Format the volume, mount it, and create a file.

- a) Enter **sudo mkfs.ext4 /dev/mapper/databk**
  - b) Verify that the file system was written.
  - c) Enter **sudo mount /dev/mapper/databk /backup/data**
  - d) Enter **echo "Encrypted" | sudo tee /backup/data/encrypt.txt**
5. Add the encrypted volume to the **/etc/crypttab** and **/etc/fstab** files.
- a) Enter **sudo bash -c "echo databk /dev/backup/databk none >> /etc/crypttab"**
  - b) Enter **sudo cat /etc/crypttab** and confirm that the line was added.

```
[student01@server01 nmap-7.70]$ sudo cat /etc/crypttab
databk /dev/backup/databk none
```

This file is similar to **/etc/fstab** and initializes encrypted storage devices at boot.

- c) Using **sudo**, open the **/etc/fstab** file in your text editor of choice.
- d) Edit the line that mounts the **/dev/backup/databk** volume to say the following:

```
/dev/mapper/databk /backup/data ext4 nofail 0 0
```

This will mount the encrypted volume after it has been unlocked. The **nofail** option indicates that the system should not report any errors if the volume is not detected.

- e) Save and close the file.

6. Reboot the machine and unlock the encrypted volume.

- a) Enter **reboot**
- b) Verify that, rather than the normal sign in screen, you are prompted to unlock the encrypted volume with your passphrase.
- c) Enter **linuxplus**
- d) Sign in as your student account.
- e) Using your preferred method, open the **/backup/data/encrypt.txt** file and verify you can read its plaintext contents.
- f) Enter **sudo bash -c "echo > /etc/crypttab"**

You're clearing this file so you won't be prompted to unlock the volume every time you reboot. You can still unlock the volume manually after you've booted into the OS.



**Note:** Encrypting a volume in this way requires physical access to the computer in order to unlock it and complete the boot process. You won't be able to SSH into the system to unlock it.

# Topic B

## Implement Identity and Access Management Methods



### EXAM OBJECTIVES COVERED

- 3.2 Given a scenario, configure and implement appropriate access and authentication methods.
- 4.3 Given a scenario, analyze and troubleshoot user issues.

One major dimension of cybersecurity is identity and access management (IAM). You'll configure various IAM solutions in order to better protect your Linux systems against unauthorized access.

### IDENTITY AND ACCESS MANAGEMENT

**Identity and access management (IAM)** is a security process that provides identity, authentication, and authorization mechanisms for users, computers, and other entities to work with organizational assets like networks, operating systems, and applications. IAM enables you to define the attributes that comprise an entity's identity, such as its purpose, function, security clearance, and more. These attributes subsequently enable access management systems to make informed decisions about whether to grant or deny an entity access, and if granted, decide what the entity has authorization to do.

For example, an individual employee may have his or her own identity in the IAM system. The employee's role in the company factors into his or her identity, like what department the employee is in, and whether or not the employee is a manager.

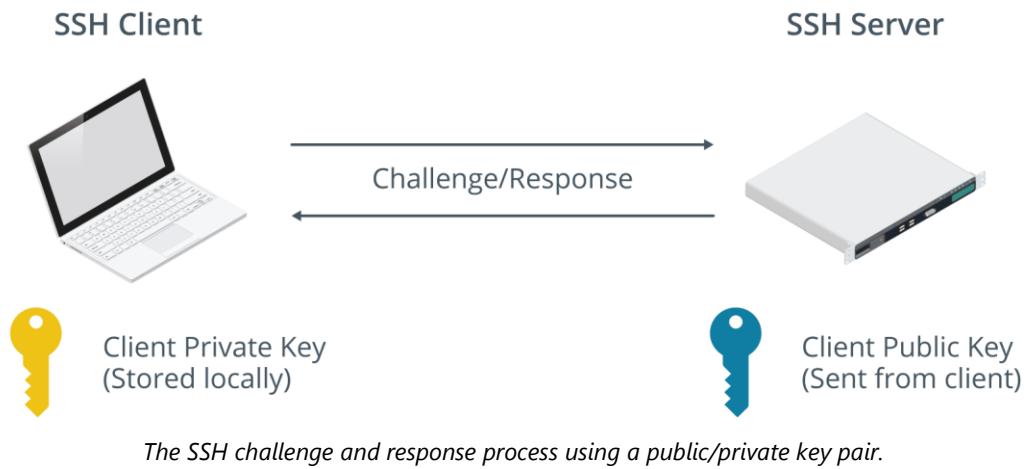
In most business environments, IAM is a crucial service for provisioning and managing access, as well as bolstering the overall security of the IT infrastructure.

### SSH AUTHENTICATION

In many distros, the default authentication method for SSH access is a password—typically the same password the local user would enter to sign in. However, this type of authentication is susceptible to various password attacks. An attacker can simply guess a poor password, like a password that is based on a common word. Or, they can automate the attack process through a brute force or dictionary attack in order to crack the password using various combinations of characters.

A more secure alternative, and one that is common in sensitive organizational environments, is to use public-key cryptography. Using public-key cryptography, the user generates a key pair—one public key, one private key. The server they are trying to remote into has a copy of the user's public key. The server presents the user with an encrypted challenge that can only be decrypted by the user's private key. If the user can successfully answer the challenge, the server can validate that they own the private key. This eliminates the risk of using a password (assuming password authentication is turned off) because the private key is virtually impossible to guess or brute force.

However, because the key is a "something you have" factor, it must be stored on a highly secure system, where the risk of it being stolen is low.



*The SSH challenge and response process using a public/private key pair.*

## SSH AUTHENTICATION FILES IN LINUX

The following is a list of files that are used to configure SSH key-based authentication in Linux:

- `~/.ssh/` —A directory that contains files related to SSH keys.
  - `id_rsa` —Contains the user's private key.
  - `id_rsa.pub`—Contains the user's public key.
  - `authorized_keys` —A file on the remote server that lists the public keys that the server accepts. In other words, the server uses this file to authenticate the client.
  - `known_hosts` —A file on the client that lists the public keys that the client accepts. In other words, the client uses this file to authenticate servers.
  - `config` —A file on the client that you can use to configure SSH connection settings, such as using an `IdentityFile` directive to associate multiple keys with specific servers.



**Note:** The `/etc/ssh/ssh_config` file is similar to `~/.ssh/config` except that it applies globally rather than to a specific user.

## SSH KEY COMMANDS

Various commands are available that you can use to work with SSH keys, including the following.

| Command                  | Used To                                                                                                                                                                                                                                          |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ssh-keygen</code>  | Generate a public/private key pair using a specified asymmetric encryption algorithm.                                                                                                                                                            |
| <code>ssh-copy-id</code> | Append the user's public keys to the remote server's <code>authorized_keys</code> file so that the server can authenticate the user's private key. The public key is sent over SSH and typically requires password authentication to be enabled. |
| <code>ssh-add</code>     | Add private key identities to the SSH key agent. If the key is protected by a password, the user only needs to enter the password once, and the agent will automatically authenticate the user.                                                  |

```
root@server01 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:YpF1pxXBVZBhiEZjbEGmq3mHym6B/D6BySvrELhdeezw
The key's randomart image is:
+---[RSA 2048]----+
| +B+=+= |
| o*= o.. |
| oo +. |
| |
| ..o S |
| ...+oo. |
| .o=+.B.. |
| +*o.o.Eo |
| o++..oo+. |
```

*Generating a public/private key pair for use in SSH.*

## THE sshd\_config FILE

The `/etc/ssh/sshd_config` file is used to configure an SSH server. Some of the settings you can configure include the following.

| Setting                              | Used To                                                                                |
|--------------------------------------|----------------------------------------------------------------------------------------|
| <code>PasswordAuthentication</code>  | Enable or disable password-based authentication.                                       |
| <code>PubkeyAuthentication</code>    | Enable or disable publickey-based authentication.                                      |
| <code>HostKey</code>                 | Reference the locations of the server's private keys.                                  |
| <code>UsePAM</code>                  | Enable or disable support for Pluggable Authentication Modules (PAM).                  |
| <code>Port</code>                    | Change the port number to bind the SSH service to.                                     |
| <code>ListenAddress</code>           | Change the IP address the SSH service should listen on.                                |
| <code>SyslogFacility</code>          | Change the logging level of SSH events.                                                |
| <code>ChrootDirectory</code>         | Reference a chroot jail path for a user.                                               |
| <code>AllowUsers, AllowGroups</code> | Enable user-specific access by allowing the specified users or groups access over SSH. |
| <code>DenyUsers, DenyGroups</code>   | Restrict the specified users or groups from accessing the server over SSH.             |
| <code>PermitRootLogin</code>         | Enable or disable the ability for the root user to log in over SSH.                    |



**Note:** The `sshd_config` file is not be confused with the `ssh_config` file mentioned earlier.

## TCP WRAPPERS

While you can deny access to specific users and groups, you can also deny connections to SSH that come from specific hosts. This is done by wrapping the SSH service in a TCP wrapper, which checks what hosts are explicitly allowed and denied before permitting the host to connect with the SSH service. You can specify hosts to allow in `/etc/hosts.allow` and hosts to deny in `/etc/hosts.deny`. The former has precedence over the latter, and is applied first. In these files you can specify hosts by their hostnames, IP addresses, network segments, etc.

For example, to deny all hosts, add the following line to `/etc/hosts.deny`:

```
sshd : ALL
```

Then, to whitelist your desired hosts, add them to `/etc/hosts.allow`:

```
sshd : 192.168.1.0/24
```

```
sshd : server01 @domain.tld
```

## PAM

**Pluggable Authentication Modules (PAM)** define the underlying framework and centralized authentication method leveraged by authentication services like Kerberos and LDAP. This provides a common mechanism for many different authentication services and applications. Authentication can therefore be streamlined within that single framework, rather than be different for each application and service.

The streamlining of authentication also benefits administrators, as PAM makes it easier for them to configure authentication policies across all applications and services on the system, as opposed to configuring policies in different formats depending on the service.

Developers can also write their own PAM modules in order to support specific authentication and authorization functions within an app.

## ACTIVE DIRECTORY

One popular implementation of LDAP is Microsoft's Active Directory® (AD). While AD is primarily implemented in Windows® environments, Linux systems can leverage pass-through authentication to forward AD credentials to PAM. For example, you can configure the System Security Services Daemon (SSSD) to cache credentials provided by AD or other external authentication mechanisms, which SSSD can then use with PAM to manage identities.

## PAM CONFIGURATION

PAM configuration files are located in the `/etc/pam.d/` directory, where each PAM-aware service or application has its own file. Each file includes directives, formatted in the following way:

```
<module interface> <control flag> <module name> <module arguments>
```

Module interfaces define functions of the authentication/authorization process contained within a module. Control flags indicate what should be done upon a success or failure of the module. The module name defines the module that the directive applies to. Module arguments are additional options you can pass into the module.

The following is an example of a password policy directive:

```
password required pam_cracklib.so retry=5
```

The module interface `password` indicates that this directive pertains to changing passwords. The `required` control flag means that the result of the module must be successful, or else the authentication process will not continue. The `pam_cracklib.so` module contains functionality that prompts a user for a

password and will test that password to see if it can be easily cracked in a dictionary attack. The **retry=5** argument gives the user five chances to fail the dictionary test.



**Note:** In some distributions, you may be able to configure PAM directly through the `/etc/pam.conf` file. The syntax of this file is similar to individual files in the `/etc/pam.d/` directory.

## MODULE INTERFACES

There are four module interfaces:

- **account** —Checks to see if a user is allowed access to something.
- **auth** —Used to verify passwords and set credentials (e.g., Kerberos tickets).
- **password** —Used to change passwords.
- **session** —Used to perform tasks in a user session that are required for access, like mounting home directories.

## CONTROL FLAGS

There are four control flags:

- **optional** —Module result is ignored.
- **required** —Module result must be successful in order to continue authentication. The user is notified when all tests in the module interfaces are finished.
- **requisite** —Same as **required**, but notifies the user immediately upon failure.
- **sufficient** —Module result is ignored upon failure.

## PASSWORD POLICIES

In addition to the prior dictionary test example, the following are some more examples of PAM password policy directives.

In the following example, the module will require that the user enter a "quality" (strong) password. Non-local users—those not found in `/etc/passwd`—are ignored:

```
password requisite pam_pwquality.so local_users_only
```

The next example enforces a password history so that users don't re-use old passwords when changing theirs. Passwords are "remembered" for 90 days:

```
password requisite pam_pwhistory.so remember=90
```

Lastly, the following example hashes the user's password using the SHA-512 algorithm. The `use_authok` argument essentially tells the module not to do any password checks, but to instead pull in the password that has already been checked by any prior modules (like quality and history)—assuming that the password has actually passed those checks:

```
password sufficient pam_unix.so sha512 use_authok
```

## USER LOCKOUTS

There are two PAM modules you can use to trigger a temporary user lockout if multiple authentication attempts fail: `pam_tally2` and `pam_faillock`. The `pam_faillock` module is recommended, as it is a newer module that improves upon `pam_tally2` by supporting user lockout when authentication is done over a screen saver.

You can place these user lockout directives in `/etc/pam.d/password-auth` and `/etc/pam.d/system-auth`

To unlock a user and reset their failure count, you can issue `pam_tally2 -r -u user`

## LDAP INTEGRATION

You can configure PAM to use LDAP by leveraging the `pam_ldap` module. Using this module, you can specify other directives that restrict what users can log in and how they can access resources. If they meet the criteria you set, the `pam_ldap` module can then authenticate the user with the LDAP service. You can add these directives to the `/etc/pam.d/common-` files.

## TTY SECURITY

Recall that, in Linux, controlling terminals are referenced by `/dev/tty#` where # is a number unique to a terminal. Each user can work with one or more terminals at a time, including the root user. However, allowing the root user to work with a terminal can become a security risk, as anyone with access to the root account can issue essentially any command on the system.

This is where the `/etc/securetty` file comes in. This file is leveraged by the `pam_securetty` module to determine what controlling terminals (`tty#`) the root user is allowed to log into. If this file does not exist, the root user can log in from any controlling terminal. If the file exists and is empty, root access is limited to single user mode and certain programs like `ssh`. Otherwise, adding the name of a controlling terminal to this file in the format `tty#` will give root access to that terminal.



**Caution:** Use `/etc/securetty` to restrict local root access only—it will not prevent root from signing through SSH. You'll need to disable that in the `sshd_config` settings on the server (`PermitRootLogin`).

## PSEUDOTERMINALS

A pseudoterminal (PTY) is an emulation of a standard controlling terminal that is used by a program. The pseudoterminal appears to other software as if it is a real terminal, but data is being input and output to the program that is emulating the terminal. For example, when you SSH into a server and enter a command, that command is sent to the pseudoterminal, which is actually controlled by the SSH service. You can enable the root user to log in to a pseudoterminal by adding a `pts/#` entry to the `/etc/securetty` file. However, this is a security risk, as it will allow insecure or malicious programs to leverage root privileges.

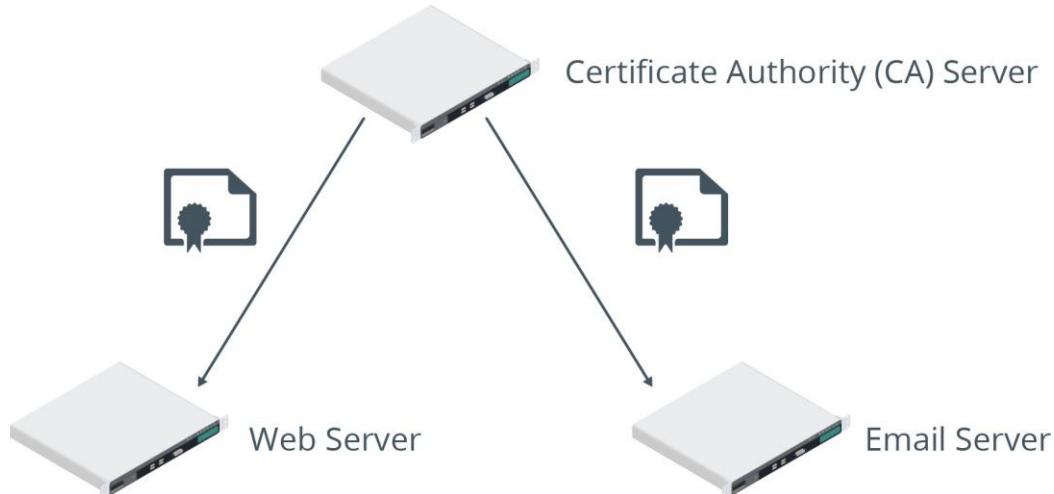
## PKI

A **public key infrastructure (PKI)** is a system that is composed of certificate authorities, certificates, software, services, and other cryptographic components, for the purpose of enabling authenticity and validation of data and entities. The PKI can be implemented in various hierarchical structures and can be publicly available or maintained privately by an organization. As its name implies, a PKI implements asymmetric cryptography for the encryption and decryption of network data, including transactions over the Internet.

## PKI COMPONENTS

There are many cryptographic components that comprise a PKI. The following table lists some of the most important of those components.

| PKI Component                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Digital signature                 | A digital signature is a message digest that has been encrypted with a user's private key. Asymmetric encryption algorithms can be used with hashing algorithms to create digital signatures. The sender creates a hashed version of the message text, and then encrypts the hash itself with the sender's private key. The encrypted hash is attached to the message as the digital signature.                                                                                                                                        |
| Digital certificate               | Digital certificates are the most fundamental component of a PKI, and the overarching task of a PKI is to manage digital certificates in a variety of ways. A digital certificate is an electronic document that associates credentials with a public key. Both users and devices can hold certificates. The certificate validates the certificate holder's identity through a digital signature and is also a way to distribute the holder's public key. In addition, a certificate contains information about the holder's identity. |
| Certificate authority (CA)        | A CA is a server that issues digital certificates for entities and maintains the associated private/public key pair. CAs sign digital certificates so that clients can validate the authenticity of certificates owned by entities. This is in contrast to a self-signed certificate—one that is owned by the same entity that signs it. In other words, the certificate does not recognize any authority, and is essentially certifying itself. Self-signed certificates require the client to trust the entity directly.             |
| Certificate signing request (CSR) | A CSR is a message sent to a CA in which an entity applies for a certificate. It typically includes information that should go into the entity's certificate, like its public key, digital signature, and other identifying information.                                                                                                                                                                                                                                                                                               |



A PKI hierarchy in which a CA issues certificates to servers.

## OpenSSL

OpenSSL is an open source implementation of the SSL/TLS protocol for securing data in transit using cryptography. On Linux, the `openssl` command is an interface into accessing a variety of OpenSSL features. It is also one of the most common tools for generating and managing components of a PKI. Using `openssl`, you can:

- Generate public and private keys.
- Generate self-signed digital certificates in various formats.
- Generate digital certificates for other entities based on CSRs.
- Calculate hash values using various functions.
- Encrypt and decrypt data using various algorithms.
- Manage keys and certificates in a CA.
- And more.

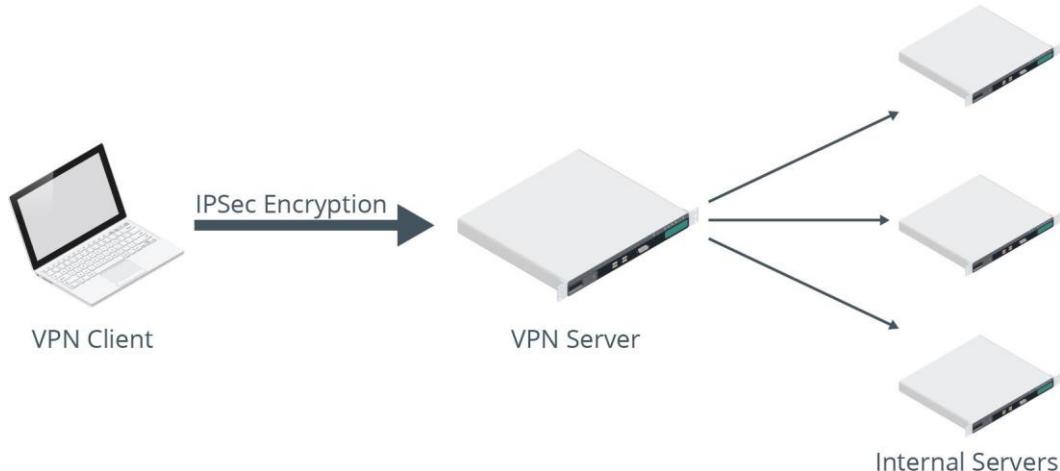
The `openssl` command can be used interactively through one of several subcommands, or you can provide these subcommands and any options non-interactively.

## SYNTAX

The syntax of the `openssl` command is `openssl [subcommand] [options]`

## VPNs AND IPSEC

In order to authenticate clients and encrypt data in transit, VPNs employ one of several tunneling protocols. One of the most prominent protocols for site-to-site connections is Internet Protocol Security (IPSec). **IPSec** is a set of open, non-proprietary standards that can be used to secure data as it travels across the network or the Internet. IPSec uses different protocols and services to provide data authenticity and integrity, anti-replay protection, non-repudiation, and protection against eavesdropping and sniffing. IPSec operates at the network layer (layer 3) of the OSI model, so the protocol is not application-dependent.



A VPN client connecting to a VPN server using IPSec encryption.

IPSec has two primary modes of operation: transport mode and tunnel mode. In transport mode, only the packet contents are encrypted, whereas the header is not. Transport mode is typically used in remote access VPNs. In tunnel mode, both the packet contents and header are encrypted. Tunnel mode is typically used in site-to-site VPNs.

Many operating systems support IPSec, including Linux. Networking devices, such as most routers, also support IPSec. Although IPSec is an industry standard, it is implemented differently in each operating system and device.

## StrongSwan

One popular utility for implementing IPSec tunnels for VPN clients is StrongSwan, available from the **strongswan** package. With StrongSwan you can set up user name and password authentication, and you can also generate digital certificates to use as a method of authentication. The main configuration file for StrongSwan is located in `/etc/strongswan/ipsec.conf` and user accounts are configurable in the `/etc/strongswan/ipsec.secrets` file.

## VPNs AND SSL/TLS

SSL/TLS is also used as a VPN authentication and encryption protocol, used primarily for remote access connections. Unlike IPSec, SSL/TLS is an application-layer (layer 7) protocol and is therefore application-dependent. One of the most popular implementations of an SSL/TLS VPN on Linux is OpenVPN. OpenVPN supports password-based, certificate-based, and smart-card based authentication mechanisms for clients. For certificate-based authentication, OpenVPN can generate self-signed certificates or leverage certificates issued from an existing CA.

OpenVPN is available through the `openvpn` package. Configuration files are typically stored in the `/etc/openvpn/` directory.

## DTLS

The **Datagram Transport Layer Security (DTLS)** protocol essentially implements SSL/TLS over datagrams (e.g., using UDP as the transport layer protocol). This means DTLS traffic is not susceptible to the same delays that TCP-oriented traffic is, particularly when TCP packets are encapsulated within a TCP connection, like in certain VPN configurations. DTLS is therefore used as an alternative VPN tunneling protocol.

OpenConnect is a popular cross-platform VPN that supports DTLS tunneling.

## ACCESS AND AUTHENTICATION TROUBLESHOOTING

In setting up remote access, you may run into issues where users cannot successfully authenticate. You need to be sure that these users have set up the proper credentials for their accounts, and that they are transmitting those credentials to the SSH, VPN, or other external authentication server. For example, in a public-key SSH authentication environment, users will need to have the correct key pair in their home directory, as well as place the SSH server in their list of known hosts. You should also check to see if their remote connection attempts are triggering a policy violation; for example, if they are trying to use a password when your configuration file permits public-key authentication only.

If remote access still fails, try signing on with the account locally to see if it's a service issue or some other networking issue. You can also test if it's a local issue by ensuring the account is active and not expired and that the password is still valid. If the account has expired or its password is otherwise invalid, you may need to reset the password using the `passwd` command. In addition, you should verify that the account is a member of the correct group(s) using the `groups` command.

If users are authenticating through an external service like a Kerberos or RADIUS/TACACS+, you should ensure the user identities are correctly configured in those services and that those services are available over the network.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

## PAM POLICY VIOLATIONS

Policy violations can also apply to PAM authentication. For example, a user may fail too many login attempts and is locked out, or the user may fail to configure their account with a suitably strong password. You should ensure that users are made aware of the policy's expectations beforehand. If the violation occurs and prevents users from correcting the issue themselves, you'll need to reset their account status. If such violations continue to occur and therefore have a negative impact on system availability, you may need to consider being more lax on the policy directives (e.g., reduce password complexity, length, or history requirements). On the other hand, you may find that lax PAM policies are leading to unauthorized users accessing resources they shouldn't, indicating that you need to tighten up your policies.

# Activity 12-3

## Discussing IAM Methods

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. You and the security team decide that, to enhance the security at Develetech, you need to use something more secure than passwords for your SSH sessions. What other authentication method can you employ to increase security?
  
2. You notice several files in your home directory under the .ssh directory. One of them is id\_rsa.pub. What is this file's purpose?
  
3. You want to set up SSH keys to better secure your SSH communications between your Linux desktop system and your Linux servers. Which command can you use to generate the public/private key pair?
  
4. OpenSSL is standard on Linux systems and is useful for securing data. What are some of its common uses?
  
5. You need to set up a VPN so that your users can securely connect to the corporate network on public Wi-Fi connections, from home networks, and from any location where they have Internet access. Which software can you investigate that supports password, certificate, and smart card authentication methods?

# Activity 12-4

## Configuring SSH Authentication Using a Key Pair

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You will work with a partner in this activity.

### SCENARIO

You want to enable your fellow administrators to remotely access servers that are physically located elsewhere. By default, the servers are already set up to accept encrypted SSH connections. Recently, however, Develetech has been the victim of several brute force password cracking attempts. Attackers have tried to gain remote access by running through various combinations of passwords. To minimize the risk of these attacks, you decide to change the authentication method that administrators will use to connect remotely. You'll have them each generate a cryptographic key pair that they'll use to prove their identities. Anyone without the key will be denied access. You'll also disable password authentication on the servers to mitigate brute force attacks.

1. Generate a public and private key pair to use with SSH authentication.
  - a) Enter `ssh-keygen`
  - b) Press **Enter** to accept the default path in which to save the key.
  - c) Enter `linuxplus` as the passphrase.  
You don't need to protect a private key with a passphrase, but doing so adds a second factor to the authentication process, and is recommended. The passphrase will decrypt the private key before it is used to solve the server's encrypted challenge.
  - d) Enter the passphrase again.
  - e) Verify that the keys were generated and saved to the home directory.  
  

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/student01/.ssh/id_rsa.
Your public key has been saved in /home/student01/.ssh/id_rsa.pub.
```
  - f) Enter `cat .ssh/id_rsa` and examine the (encrypted) private key.  
This is the key you'll use to validate the SSH server's encrypted challenge.
  - g) Enter `cat .ssh/id_rsa.pub` and examine the public key.  
The server needs to install this public key once. The server will use this public key to verify the authenticity of the private key.
2. Copy your public key to your partner's server.
  - a) Enter `ssh-copy-id student##@<partner's IP address or hostname>`



**Caution:** The student### number should be your partner's number, not yours. This is because they don't have your specific student account on their system, so you have to use theirs.

For example: **ssh-copy-id student01@10.50.1.101**

- b) Enter **yes** to accept the authenticity of your partner's host machine.
  - c) When prompted for a password, enter **Pa22w0rd**
  - d) Verify that the key was added.
3. Verify that your partner's public key was added to your server.
- a) Wait for your partner to finish copying their key to your server.
  - b) Enter **cat .ssh/authorized\_keys** and verify that your partner's key was added.

```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDGFMf3V+W7hbWbjnNJGvqzXoYX3CcxoGGV0
sf0i6ML7f1S6fHW6390XdjSInXu0hLb2LNtNnh+6VaST36oLqB5NHrUUiZYtzS3dX82+AwXpX
s402AX+d5ByGhnmcNHJKWmeYkdiqwt7w03sld+/wWmB4i0c7K4pQTkjOSAtkAeRTG6XDZYfJM
DFwYS6pRN0vZ7CumARFxRwLtYaDcdbNCxaz1fJroEsSh1H+79jF/e8LtfRr27P7o7MZLHsJ5
E6+vNpjKmyFRNtGjP0byYK0h3EM9n1yHy3y3d2mrvhbDqUQE0Bko7kEo2o3EvnrKQ7oytPV2
c0ZmRNYca5bHS39 student02@server02
```

Any public keys added to this file are considered authorized and will be used in SSH authentication. If you wanted to authenticate other users, you could have them generate a unique key pair and then add their public key to this file as well.

4. Authenticate with your partner's SSH server using your private key.
- a) Enter **SSH student##@<partner IP address or hostname>**



**Caution:** Remember, use your partner's student number.

- b) When prompted, type (but don't press **Enter**) **linuxplus** as the passphrase to unlock your private key.
- c) Check the **Automatically unlock this key whenever I'm logged in** check box.
- d) Select **Unlock**.
- e) Verify that you are signed in to your partner's server as their student account.



**Note:** If you get an "Authentication failed" message, enter the **ssh** command again.

You've successfully authenticated with your partner's SSH server.

5. For added security, disable password authentication.
- a) Enter **exit** to close your SSH session and return to your local login.
  - b) Using **Sudo**, open the **/etc/ssh/sshd\_config** file in your desired text editor.
  - c) Scroll down until you get to the **PasswordAuthentication yes** line.
  - d) Change **yes** to **no** and then save and quit the file.
  - e) Enter **Sudo systemctl restart sshd**
  - f) Wait for your partner to disable password authentication on their server.
  - g) Enter **SSH ariley@<partner IP address or hostname>**
  - h) Verify that permission is denied.

You have no private key for this account, and the server isn't accepting passwords.

# Topic C

## Configure SELinux or AppArmor



### EXAM OBJECTIVES COVERED

- 3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.
- 4.3 Given a scenario, analyze and troubleshoot user issues.
- 4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

Several Linux distributions provide an additional layer of security on top of the operating system. That layer is usually implemented in the form of SELinux or AppArmor, both of which can further harden your Linux systems against malicious or careless use.

### CONTEXT-BASED PERMISSIONS

**Context-based permissions** describe multiple types of information about processes and files that are used in combination to make decisions related to access control. In other words, the permission scheme defines various properties for a file or process, and uses those properties together, rather than in isolation, to determine whether to grant or deny access. This makes context-based permissions more advanced than the default scheme of granting a user or group access to a file directly.

In Linux, there are two main context-based permission schemes available: SELinux and AppArmor.

### MAC

**Mandatory access control (MAC)** is a model in which access is controlled by comparing an object's security designation and a subject's (users or other entities) security clearance. Objects such as files and other resources are assigned security labels of varying levels, depending on the object's sensitivity. Subjects are assigned a security level or clearance, and when they try to access an object, their clearance level must correspond to the object's security level. If there is a match, the subject can access the object; if there is no match, the subject is denied access.

Both context-based permissions schemes in Linux leverage MAC. This differs from the default scheme in Linux, discretionary access control (DAC), in which each object has a list of entities that are allowed to access it, and which the object owner can change directly.

### SELinux

**Security-Enhanced Linux (SELinux)** is the default context-based permissions scheme provided with CentOS and Red Hat Enterprise Linux, and is optionally available on other distributions. It was developed by the U.S. National Security Agency (NSA). It provides additional file system and network security so that unauthorized processes cannot access or tamper with data, bypass security mechanisms, violate security policies, or execute untrustworthy programs.

SELinux enforces MAC on processes and resources and enables information to be classified and protected based on its confidentiality and integrity requirements. This helps mitigate the damage caused to information by malicious applications and users.

## SELinux CONTEXTS

SELinux defines three main contexts for each file and process. When you list an object's contexts, each one is delineated by a colon.

- **User:** This context defines what users can access the object. Note that this *does not* refer to Linux system users, but distinct SELinux users. Each Linux system user is mapped to one of these SELinux user values. Different distributions provide different users, but common ones include:
  - `unconfined_u` —All users.
  - `user_u` —Unprivileged users.
  - `sysadm_u` —System administrators.
  - `root` —The root user.
- **Role:** This context defines what roles can access the object. SELinux users are authorized to be in roles. Roles are typically used to permit or deny users access to domains, which apply to processes. The `object_r` role applies to files and directories.
- **Type:** This context is the "label" portion of MAC, and is perhaps the most important context for fine-grained access control. It is a way of grouping objects together that have similar security requirements or characteristics. The word "type" usually applies to files and directories, whereas a "domain" is just a type that applies to processes. For example, `ssh_t` is the domain for the SSH process.

```
root@server01 ~]# ls -lZ file1.txt anaconda-ks.cfg
-rwxrwxr-x. root root system_u:object_r:admin_home_t:s0 anaconda-ks.cfg
-rwxrwxr-x. root root unconfined_u:object_r:admin_home_t:s0 file1.txt
```

The diagram illustrates the SELinux context structure for the files listed in the terminal. The context is represented as `user:role:type`. In the output, `system_u` is the User, `admin_home` is the Role, and `s0` is the Type. The `unconfined_u` entry is also shown, likely representing a different context or a specific state.

An example of a file's SELinux contexts.

## MULTI-LEVEL SECURITY

Multi-level security (MLS) is an optional feature of SELinux that enables a fourth context, called a level, to describe the sensitivity level and/or category of an object. This enables you to further fine-tune and constrain access even when the main three contexts are fulfilled.

## SELinux MODES

SELinux has three different modes. Each mode configures the overall implementation of SELinux on the system.

| Mode     | Description                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------|
| Disabled | In this mode, SELinux is turned off. So, MAC will not be implemented and the default DAC method will be prevalent. |

| Mode       | Description                                                                                                                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enforcing  | In this mode, all the SELinux security policies are enforced. Therefore, processes cannot violate the security policies.                                                                                                         |
| Permissive | In this mode, SELinux is enabled, but the security policies are not enforced. So, processes can bypass the security policies. However, when a security violation occurs, it is logged and a warning message is sent to the user. |

## SELinux POLICIES

An SELinux security policy defines access parameters for every process and resource on the system. It enforces rules for allowing or denying different domains and types to access each other. Configuration files and policy source files located in the `/etc/selinux/` directory can be configured by the root user.

Each policy is categorized as either **targeted** or **strict**. According to a targeted policy, except the targeted subjects and objects, all other subjects and objects will run in an unconfined environment. The untargeted subjects and objects will operate on the DAC method and the targeted ones will operate on the MAC method. A targeted policy is enabled by default.

A strict policy is the opposite of a targeted policy, where every subject and object of the system is enforced to operate on the MAC method.

## SELinux COMMANDS

The following table describes some of the major commands that you can use to configure an SELinux environment.

| Command                 | Used To                                                                                                                                                                                                      |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>semanage</code>   | Configure SELinux policies.                                                                                                                                                                                  |
| <code>sestatus</code>   | Get the status of SELinux, including its current mode, policy type, and mount point.                                                                                                                         |
| <code>getenforce</code> | Display which mode SELinux is running in.                                                                                                                                                                    |
| <code>setenforce</code> | Change which mode SELinux runs in. You can issue <code>setenforce 1</code> to enable enforcing mode and <code>setenforce 0</code> to enable permissive mode.                                                 |
| <code>getsebool</code>  | Display the on/off status of SELinux boolean values. Boolean values enable you to change policy configurations at runtime without actually writing the policy directly.                                      |
| <code>setsebool</code>  | Change the on/off status of an SELinux boolean value.                                                                                                                                                        |
| <code>ls -Z</code>      | List directory contents along with each object's security context. You can check the context of specific objects by issuing <code>ls -Z {file or directory name}</code>                                      |
| <code>ps -Z</code>      | List running processes along with each process's security context. You can check the context of specific processes by issuing <code>ps -Z {PID}</code>                                                       |
| <code>chcon</code>      | Change the security context of a file. The basic syntax is <code>chcon {-u -r -t} {context value} {file or directory name}</code> where <code>{-u -r -t}</code> refers to user, role, or type, respectively. |

| Command    | Used To                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| restorecon | Restore the default security context of one or more files.<br>You restore objects by issuing <b>restorecon {file or directory name}</b> |

Status and info

```
root@server01:~]# sestatus
SELinux status: enabled
SELinuxfs mount: /sys/fs/selinux
SELinux root directory: /etc/selinux
Loaded policy name: targeted
Current mode: permissive
Mode from config file: enforcing
Policy MLS status: enabled
Policy deny_unknown status: allowed
Max kernel policy version: 31
[root@server01 ~]#
```

Checking the status of SELinux.

## SELinux VIOLATIONS

Violations occur when SELinux denies a subject access to an object based on policy. This is typically because the subject has insufficient privileges and is therefore not authorized by the SELinux environment to perform a particular action. Although violations are expected under certain circumstances, they can sometimes affect users and processes that should be able to access an object.

One way to diagnose and troubleshoot unexpected violations is by using the **sealert** command with the **-a** option and the audit log provided as an argument. For example:

**sealert -a /var/log/audit/audit.log**

This will display all of the policy violations that have occurred, along with detailed information about each violation, including the timestamp, log type, permission requested, names of the process and the target it tried to access, security contexts of both, and more.

The output of **sealert** can be difficult to parse, so you can use the **audit2why** command to translate an event into a more human-friendly format that explains why a violation occurred. You can redirect the entire log to this command. If you only want to analyze one or a few events, you can grep an identifier that is unique to an event, like its timestamp, and then pipe that to the **audit2why** command.

## AppArmor

**AppArmor** is an alternative context-based permissions scheme and MAC implementation for Linux. Whereas SELinux is more commonly associated with RHEL, AppArmor is packaged with Debian-based and SUSE Linux distros. AppArmor provides the same fundamental service as SELinux, but its approach is different in many

significant ways. Perhaps the most overarching difference is that SELinux is very complex and often difficult to configure, whereas AppArmor was designed to be much simpler.

Functionally, the main difference is that AppArmor works with file system objects based on paths, whereas SELinux references inodes directly. These paths are referenced in flat configuration files, or profiles, that AppArmor uses to determine how to control access. This also means that there are no types or domains in AppArmor, only these profiles.

## AppArmor PROFILES

Each executable can have an associated AppArmor profile. Profiles are located in the `/etc/apparmor.d/` directory. Within this directory are several text files that are named in a `path.binary` format. For example, the `/bin/dig` command binary's AppArmor configuration file would be located at `/etc/apparmor.d/ bin.dig`.

Within a profile, you can configure two main types of rules: capabilities and path entries. Capabilities provide the executable in question access to some sort of system functionality. For example, the `net_bind_service` capability enables the executable to bind to a well-known TCP/IP port (port numbers below 1024).

Path entries enable the executable to access a specific file on the file system. As the name suggests, you reference the files by their paths. After the path you specify what permissions you want to grant to this executable for the files. There are several possible permissions, including `r` for read, `w` for write, `ux` for unconfined execute (file being accessed doesn't have a profile), `l` for link, and so on.

## AppArmor MODES

Each profile operates in one of two modes: complain and enforce. In complain mode, profile violations are logged but not prevented. In enforce mode, profile violations are both logged and prevented.

## AppArmor TUNABLES

Tunables enable you to configure AppArmor functionality without directly modifying profiles. For example, profiles may reference a common object or path using a variable name, like `@{HOME}` to refer to the user's home directory. If the user's home directory is not in the default location, you can adjust the appropriate tunable file to account for this. Tunable files are located in the `/etc/apparmor.d/tunables/` directory.

## AppArmor COMMANDS

The following table describes some of the major commands that you can use to configure an AppArmor environment.

| Command                      | Used To                                                                                                        |
|------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>apparmor_status</code> | Display the current status of AppArmor profiles.                                                               |
| <code>aa-complain</code>     | Place a profile in complain mode. The basic syntax is <code>aa-complain {path to profile}</code>               |
| <code>aa-enforce</code>      | Place a profile in enforce mode. The basic syntax is <code>aa-enforce {path to profile}</code>                 |
| <code>aa-disable</code>      | Disable a profile, unloading it from the kernel. The basic syntax is <code>aa-disable {path to profile}</code> |

| Command       | Used To                                                                              |
|---------------|--------------------------------------------------------------------------------------|
| aa-unconfined | List processes with open network sockets that don't have an AppArmor profile loaded. |



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 12-5

## Discussing SELinux and AppArmor

### SCENARIO

Answer the following questions to check your understanding of the topic.

---

- 1. You are in the testing phase of implementing some new services and SELinux keeps blocking external users from testing. How can you allow access without disabling SELinux or spending a lot of time setting up exceptions to allow all the ports required for testing?**
  
  - 2. You SSH into a Linux system that only allows SSH access. The system is a web server and you've checked to see everything is working properly. However, no one can connect to the web server with a web browser. You suspect SELinux is in enforcing mode and blocking access. How can you check?**
  
  - 3. You are researching a possible breach attempt on one of your SELinux-protected systems. Which command can you use to check the SELinux audit log?**
  
  - 4. You are deploying four Debian-based Linux servers and you do not want to use SELinux for access control. What is the most common alternative to SELinux for Debian-based systems?**
  
  - 5. You want to set some very restrictive protection on certain executables. AppArmor gives you that capability. Which mode do you use for the most restrictive profiles for your executables?**
-

# Activity 12-6

## Configuring SELinux

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. Previously, you installed Apache HTTP Server.

### SCENARIO

The Apache web server you installed is serving its purpose, but the team would like to organize the server's files in a more descriptive way than the default `/var/www/html` directory. Also, the system will eventually run multiple web apps, each in a different path. For now, you need to place all of the web server files in a new `/var/develweb` directory. Even though you apply the correct standard permissions and configure Apache to allow access, SELinux will prevent the `httpd` service from reading files in this new directory. So, you'll configure SELinux as needed to make sure the web server is operational.

1. Check the status of SELinux.
  - a) Enter `sestatus`
  - b) Verify that SELinux is enabled and in enforcing mode.
  
2. Create a new directory to hold the web server files.
  - a) Enter `sudo mkdir /var/develweb`
  - b) Enter `Sudo bash -c "echo This is a test > /var/develweb/test.html"`
  - c) Enter `ls -al /var/develweb`
  - d) Verify that all users have read and execute permissions to this directory, and that all users have read permissions to the file you just created.
  
3. Configure Apache settings to use the new path as the document root and enable access.
  - a) Using `sudo`, open the `/etc/httpd/conf/httpd.conf` in the text editor of your choice.
  - b) Enter `/DocumentRoot` to search for the appropriate field.
  - c) Change this field to the following:  
`DocumentRoot "/var/develweb"`
  - d) Below this, look for the `<Directory "/var/www">` line and change it to the following:  
`<Directory "/var/develweb">`
  - e) Save and quit the file.
  - f) Enter `sudo systemctl restart httpd`

4. Attempt to view the web page, and troubleshoot any issues.
  - a) From the desktop menu, select **Applications**→**Favorites**→**Firefox Web Browser**.
  - b) Navigate to **http://127.0.0.1/test.html**.
  - c) Verify that you are presented with a **Forbidden** message.  
The message elaborates by saying you don't have permission to access the HTML file. An SELinux alert will also pop up on the desktop, but will be minimized after a few seconds.
  - d) If the SELinux alert is still open, hover over it and select **Show**. If the alert disappears, select the icon to the left of the clock  to open the same **SELinux Alert Browser** dialog box.
  - e) Select **Troubleshoot**.
  - f) Under **If you were trying to**, select the first option.
  - g) Verify that the proposed solution involves changing the context label on the new directory path.  
The troubleshooter mentions many possible file types, but there is a way to narrow down which one you need to use.
  - h) Close this dialog box.
5. Verify the required SELinux context and the current one applied to the new path.
  - a) At a terminal, enter **ls -Z /var/www**  
Retrieving SELinux context information from the default document root should provide you with what you need.
  - b) Verify that the "type" context for the **html** directory is **httpd\_sys\_content\_t**  
This is the context you need to apply to your new directory to allow Apache to access the files.
  - c) Enter **ls -aZ /var/develweb** and verify that the "type" context for your new directory is **var\_t**  
This new directory inherited the default context of its container directory (**/var**).
6. Apply the appropriate context to the new web server directory.
  - a) Enter the following:  
**sudo semanage fcontext -a -t httpd\_sys\_content\_t "/var/develweb(/.\*)?"**  
This adds the provided context for the **/var/develweb** directory. The symbols **(.)?** at the end use a regular expression to apply this context to all subdirectories and files within this directory.
  - b) Enter **ls -aZ /var/develweb** and verify that the directory still doesn't have the correct context.
  - c) Enter **sudo restorecon -Rv /var/develweb**
  - d) Check the directory's contexts again and verify that it now shows **httpd\_sys\_content\_t**
7. Confirm that you can now access the web page.
  - a) Switch back to Firefox and refresh the page.
  - b) Verify that the test page is displayed.
  - c) Close the browser.

# Topic D

## Configure Firewalls



### EXAM OBJECTIVES COVERED

- 3.5 Given a scenario, implement and configure Linux firewalls.
- 4.1 Given a scenario, analyze system properties and remediate accordingly.
- 4.4 Given a scenario, analyze and troubleshoot application and hardware issues.

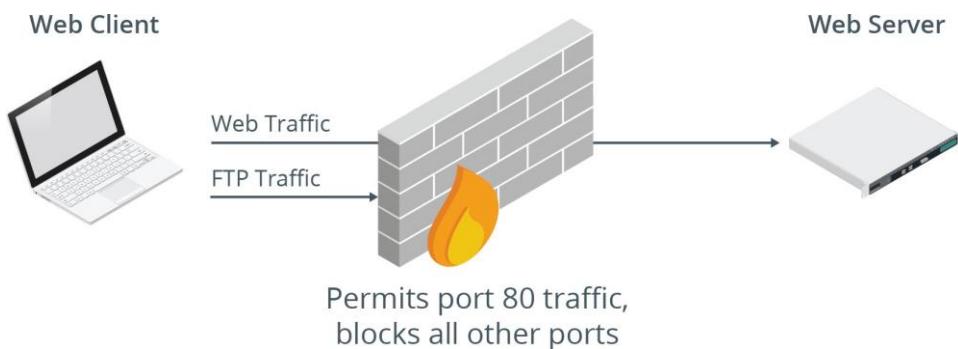
You've implemented IAM and context-based permissions, but there is still a major source of risk on your Linux systems: the network. By configuring a firewall, you'll help keep unwanted network traffic out while allowing legitimate traffic in.

### FIREWALL

A **firewall** is a software program or a hardware device that protects a system or a network from unauthorized access by blocking unwanted traffic. A firewall can allow or deny incoming and outgoing traffic based on a set of rules that are either explicitly configured by an administrator or which are active by default. Firewalls often provide logging features and alerts that track security problems and report them to the administrator.

There are three main generations of firewalls:

- **Packet filters (first generation):** These firewalls make decisions based on rules that correspond to one or more network packet attributes. These rules appear in the form of an access control list (ACL). Packet filtering firewalls are also called *stateless* firewalls because they can only inspect a packet in isolation, and cannot determine what has come before that packet that might provide valuable context.
- **Stateful firewalls (second generation):** In contrast to packet filters, stateful firewalls *can* identify past traffic that is related to a packet. This means that a stateful firewall can view the entire conversation of a transmission, such as the three-way TCP/IP handshake. Stateful firewalls can therefore make more informed decisions about what traffic to deny and what to allow.
- **Application-layer firewalls (third generation):** These firewalls can inspect the contents of application-layer traffic (e.g., protocols like HTTP and FTP) and make decisions based on these contents. An application-layer firewall can detect attempts to bypass traditional filtering and stateful inspection that leverage known software exploits.



*A packet filtering firewall.*

## ACL FEATURES

A stateless firewall's ACL can allow or deny packets based on various factors. Those factors include:

- Source IP address
- Destination IP address
- Source TCP or UDP port
- Destination TCP or UDP port
- TCP or UDP protocol used

For example, one of the most common ways to configure an ACL is to deny incoming traffic that uses a port that the network or its systems do not need to use. You might configure the ACL to block incoming traffic on port 21 (FTP) if it is bound for a system that doesn't host an FTP server.

Once the firewall matches traffic to a rule, it needs to perform one of the following actions:

- **Accept:** The traffic is allowed through the firewall and sent on to its destination.
- **Reject:** The traffic is blocked at the firewall and the firewall notifies the sender.
- **Drop:** The traffic is blocked at the firewall and the firewall *does not* notify the sender.

In addition, you can configure the firewall to log any of the previous actions. These logs typically include information about the packet, like its source and destination, as well as timestamps and other useful data.

## THE iptables TOOL

The **iptables** tool enables you to manage packet filtering as well as stateful firewall functionality within Linux through various tables. Each table applies to a certain context and consists of rule sets, called chains, that the table uses to implement the firewall. A packet is compared to the first rule in the appropriate chain, and if it does not match that rule, it is compared to the next rule in the chain, and so on. If the packet matches a rule, it can either be evaluated by a new chain or have one of three actions applied to it: **ACCEPT**, **DROP**, or **RETURN** (skip to next rule in previous chain).

```
[root@server01 ~]# iptables -L IN_dmz
Chain IN_dmz (1 references)
target prot opt source destination
IN_dmz_log all -- anywhere anywhere
IN_dmz_deny all -- anywhere anywhere
IN_dmz_allow all -- anywhere anywhere
ACCEPT icmp -- anywhere anywhere
```

**Rules**: The title of the terminal window.

**Chain**: The name of the chain being listed, shown in the terminal output as "Chain IN\_dmz".

**Action**: The action taken when a packet matches a rule, shown in the terminal output as "target".

**Protocol**: The type of protocol handled by the rule, shown in the terminal output as "prot".

*Listing the rules in a firewall chain.*

Each table has one or more built-in chains, but you can also define your own chains as desired.

## SYNTAX

The syntax of the **iptables** command is **iptables [options] [-t table] [commands] {chain/rule specification}**

## DEFAULT TABLES

There are five default tables that may be active depending on how the kernel is configured:

- **filter**—The default table used for typical packet filtering functionality.
- **nat**—Used to implement Network Address Translation (NAT) rules.
- **mangle**—Used to alter packets' TCP/IP headers.
- **raw**—Used to configure exceptions for packets involved in connection tracking.
- **security**—Used to mark packets with SELinux security contexts.

## PERSISTENCE

By default, rules set with **iptables** will be lost on reboot. In CentOS/RHEL, you can install the **iptables-services** package and issue the **service iptables save** command to ensure your changes persist. For Debian-based distros, you can install the **iptables-persistent** package. After installation, you'll be asked to confirm that you want your current rules to persist. The **iptables-persistent** service will then automatically run at boot and load your rules.

## LOGGING

You can enable logging for **iptables** rules by including the **LOG** action. In the following example, all dropped packets are being logged:

```
iptables -N LOGCHN
iptables -I INPUT -j LOGCHN
iptables -I LOGCHN -j LOG
iptables -I LOGCHN -j DROP
```

The first line creates a new chain called **LOGCHN**. The second line ensures all incoming packets not already processed by any prior rules will "jump" to the **LOGCHN** chain. The third line logs all packets that reach this chain, and the fourth line performs the actual dropping of packets. You can also substitute **ACCEPT** for **DROP** if you only want to log accepted packets.

Events for **iptables** are typically written to the **/var/log/messages** or **/var/log/kern.log** files.

## UFW

The **Uncomplicated Firewall (UFW)** is a firewall management tool that makes it easier to configure the **iptables** service. UFW originated with Ubuntu® but can be downloaded and installed on other distributions. It is primarily useful for home users who don't have experience with the intricacies of firewall configuration.

The **ufw** command enables you to work with the command-line interface. For example, the following commands set up an allow rule for HTTP, turn on logging, and enable the firewall. This automatically creates a default deny configuration for incoming traffic—in other words, everything without an explicit allow rule is dropped:

```
ufw allow http/tcp
ufw logging on
ufw enable
```

## SYNTAX

The syntax of the `ufw` command is `ufw [options] {action}`

## ADVANCED CONFIGURATION

If you want to use UFW to employ a more complex firewall configuration, you'll need to edit text files rather than use the `ufw` command. The `/etc/default/ufw` file is used to configure high-level settings like policy defaults and kernel module usage.

More granular configuration files are found in the `/etc/ufw/` directory. You can edit these files to control when rules are applied, when customizations are run with respect to the `ufw` command, and more.

For example, UFW defaults to accepting outgoing traffic through the firewall. You change this behavior by specifying a different policy directive in `/etc/default/ ufw`:

```
DEFAULT_OUTPUT_POLICY="DROP"
```

As a more granular example, you can configure `/etc/ufw/applications.d/ myapp` to instruct UFW to recognize your app and its port/protocol information:

```
[My App]
title=My App
description=My custom application
ports=23534/tcp
```

## THE firewalld SERVICE

The firewall daemon (`firewalld`) is used to dynamically manage a firewall without requiring the firewall to restart upon modification. It is an alternative to `iptables` and uses zones and services rather than chains and rules.

**Firewall zones** are the rule sets that can apply to specific network resources, like a network interface. You'd typically place resources in a zone to group them with resources that have similar security requirements or similar levels of trust. There are various default zones, each with different levels of trust. For example, the zone with the lowest level of trust is called `drop` and it immediately drops all incoming connections. Firewall services are the rules that apply to specific services that operate within a zone. For example, you can add a service like HTTP to the `dmz` zone to allow incoming connections from untrusted networks like the Internet, while denying outgoing access to the rest of the network.

## THE firewall-cmd COMMAND

The `firewall-cmd` command enables you to configure `firewalld` by querying, adding, modifying, and deleting zones and services as desired. Because `firewalld` is the default firewall service for many Linux distributions, including Red Hat® Enterprise Linux® and CentOS®, you will be using the `firewall-cmd` command regularly. The command includes options to identify which zone and which interface you want to configure, as well as the ability to permit services by name or by port number.

```
root@server01 ~]# firewall-cmd --list-all --zone=dmz
dmz (active)
target: default
icmp-block-inversion: no
interfaces: enp3s0 ← Interface applied to zone
sources:
services: ssh http https ← Services applied to zone
ports: 7743/tcp 601/tcp ← Ports applied to zone
protocols:
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

*Listing the configurations for a specific firewalld zone.*

## SYNTAX

### X

The syntax of the **firewall-cmd** command is **firewall-cmd [options]**

## firewall-cmd COMMAND EXAMPLES

The following are some common examples of using the **firewall-cmd** command:

- **firewall-cmd --get-zones** — list all available **firewalld** zones.
- **firewall-cmd --zone=dmz --list-all** — list all details of the **dmz** zone, including the interfaces, ports, services, protocols, and more that the zone applies to.
- **firewall-cmd --zone=dmz --change-interface=<device ID>** — add the specified interface to the **dmz** zone.
- **firewall-cmd --zone=dmz --add-service=http** — add the HTTP service to the **dmz** zone.
- **firewall-cmd --zone=dmz --add-port=21/tcp** — add TCP port 21 (FTP) to the **dmz** zone.
- **firewall-cmd --zone=dmz --remove-service=http** — remove the HTTP service from the **dmz** zone.
- **firewall-cmd --zone=dmz --remove-port=21/tcp** — remove TCP port 21 (FTP) from the **dmz** zone.
- **firewall-cmd --reload** — reload the zone's configuration.

## PERSISTENCE

Like **iptables**, **firewalld** does not persist its changes by default. This is called runtime mode. You must commit a change with the **--permanent** option for it to persist upon restart of the daemon.

## NETFILTER

**Netfilter** is a Linux kernel framework that handles packets that traverse a network interface. Some of the major services it provides are packet filtering, NAT, and

connection tracking. Netfilter supports the configuration of these services by providing hooks into the kernel's network stack. Every packet that traverses the network interface will be "caught" by these hooks. User space programs that are registered with the relevant hooks are able to interact with the packets on the hooks.

The **iptables** tool is closely integrated with Netfilter. It is able to allow, drop, and perform other firewall actions because it can interact with packets that are on Netfilter hooks. Both UFW and **firewalld** call **iptables** in some capacity, so they likewise rely on Netfilter.

## IP FORWARDING

**IP forwarding** is the Linux kernel implementation of network routing functionality. It enables incoming traffic on one network interface to be forwarded to another network interface. IP forwarding is therefore only useful on systems that have multiple interfaces, particularly systems that act as routers or gateways for other systems in the network.

IP forwarding is often used in conjunction with **iptables** firewall configuration. For example, say you have a Linux host acting as a router. It has one public, Internet-facing interface; and one private, internal-facing interface. You also have a separate web server with only a private interface. You want traffic from the Internet to pass through your router, and only be forwarded on to the internal network if the traffic uses ports 80 or 443 (HTTP/S). You can do this by configuring a **FORWARD** chain that will allow traffic on ports 80 and 443 to be forwarded on to the private interface, while preventing any other kind of traffic from being forwarded.

In order to leverage IP forwarding, you must first enable it in the kernel. This is as simple as altering a single value in the appropriate file:

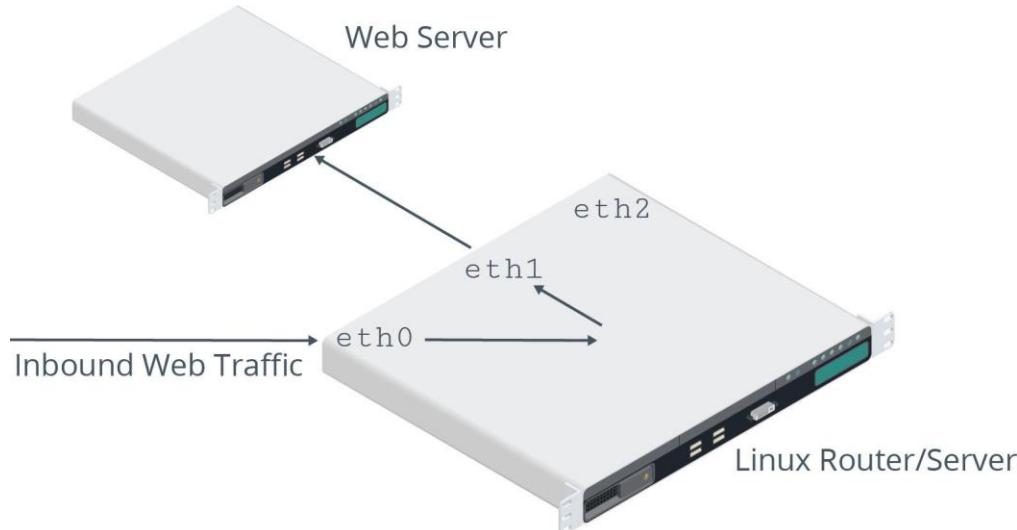
```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Or, for IPv6 traffic:

```
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
```



**Note:** To make this change persist, you must enable the appropriate setting in the `/etc/sysctl.conf` file.



A Linux system receiving traffic on one NIC and forwarding that traffic to another NIC.

## IP SETS

**IP sets** are stored collections of IP addresses, network ranges, MAC addresses, port numbers, and network interface names. The **iptables** tool can leverage IP sets for more efficient rule matching. For example, let's say you want to drop traffic that originates from one of several IP address ranges that you know to be malicious.

Instead of configuring rules for each range in **iptables** directly, you can create an IP set and then reference that set in an **iptables** rule. This makes your rule sets dynamic and therefore easier to configure; whenever you need to add or swap out network identifiers that are handled by the firewall, you simply change the IP set.

The **ipset** command enables you to create and modify IP sets. First you need to set a name, storage method, and data type for your set, such as:

```
ipset create range_set hash:net
```

In this case, **range\_set** is the name, **hash** is the storage method, and **net** is the data type. Then, you can add the ranges to the set:

```
ipset add range_set 178.137.87.0/24
ipset add range_set 46.148.22.0/24
```

Then, you use **iptables** to configure a rule to drop traffic whose source matches the ranges in this set:

```
iptables -I INPUT -m set --match-set range_set src -j DROP
```

Alternatively, to drop traffic whose destination matches the set:

```
iptables -I OUTPUT -m set --match-set range_set dst -j DROP
```

## SYNTAX

The syntax of the **ipset** command is **ipset [options] {command}**

## TROUBLESHOOTING

The **ipset** tool can also be used when troubleshooting the **iptables** firewall. For example, you can use the **test** subcommand to test whether or not an entry exists:

```
ipset test range_set 178.137.87.5
```

If the firewall still isn't handling the IP address ranges as expected, you can list the rules that are using the relevant set:

```
iptables -L | grep range_set
```

Even if the rules are using your set, keep in mind that the rules are processed in order; the unexpected behavior may be due to how these rules flow in the table.

## FIREWALL CONFIGURATION FOR APPLICATIONS

As you know, network services and applications require the use of a port to establish a connection endpoint. Most common protocols have a dedicated port number as assigned by the Internet Assigned Numbers Authority (IANA). However, you may need to run a custom or uncommon application that requires network access, and this application may not have a standardized port number. In that case, you'll need to choose a port number and associate it with your application, then open that port at the firewall.

Linux keeps a database of services and their corresponding port numbers in the **/etc/services** file. This file enables services to, by default, attempt to bind to their corresponding port when activated. The format of each entry is:

```
service-name port/protocol [aliases] [# comment]
```

So, to map an application called my-app to port number 55111, you'd add the following line:

```
my-app 55111/tcp # My custom app
```

Whenever my-app is started, it will attempt to bind to port 55111 and start listening on that port. So, you'd use a firewall service like **iptables** or **firewalld** to allow traffic bound for port 55111 on the network.

## TRUSTED PORTS

**Trusted ports**, also called **privileged ports**, are ports in the well-known range (0– 1023). In Linux, if a process is to start listening on a trusted port, or to establish a remote connection from a trusted port, it must have superuser privileges. This helps the other side of the connection confirm that the service they are connecting to is trusted. This is especially important when it comes to FTP, HTTP, SSH, and other common protocols that involve the transfer of data. Because many servers use these protocols, you'll need to ensure that their ports are open in the firewall.

## FIREWALL TROUBLESHOOTING

As you configure and implement a firewall, you may run into a common issue: The firewall blocks traffic that it shouldn't. The following table lists some potential causes and solutions associated with this issue.

| Cause             | Solution                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Blocked ports     | Check your firewall's rule set to ensure that it is not overtly blocking a port that your system needs in order to forward outgoing traffic. Likewise, your firewall is likely in default deny mode for incoming connections, so be sure to create an explicit rule that allows traffic on the port you need.                                                                                                                                                                                                                                                 |
| Blocked protocols | Even though you may be correctly allowing a port, you may not have configured it for the correct protocol. Ensure that you are opening up the port on either TCP, UDP, or both— depending on what transport protocol the connection requires. Also ensure that, for application-layer firewalls, they are not configured to drop packets whose contents match a specific application-layer protocol (e.g., HTTP, FTP, SSH).                                                                                                                                   |
| Restrictive ACLs  | The factors that an ACL uses to allow or deny traffic can be used in conjunction; for example, you can configure an ACL to only block a specific source port if its source matches a known IP address and if it is headed for a specific destination port. Granular filtering like this can be very useful, but it can also become complex and has a higher potential for false positives. Configure your ACL rules to be as straightforward as your needs allow and don't get carried away with trying to write a granular rule for every possible scenario. |

## IPSs

An **intrusion prevention system (IPS)** is a security appliance that monitors and evaluates a system for signs of attacks in progress, and can actively block traffic that it determines is malicious. IPSs are similar in purpose to firewalls in that both can stop unwanted traffic from entering a network. However, they have some key differences.

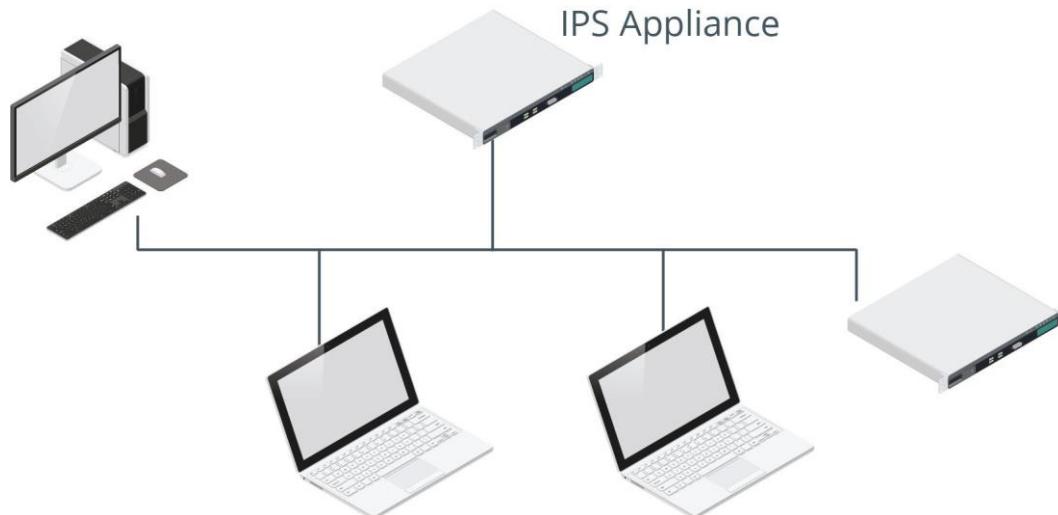
A firewall is like a security guard who lets guests into a building based on whether or not they match predefined rules. An IPS is more akin to a second security guard inside a building. Even if the outside guard (the firewall) lets someone in, the inside guard (the

IPS) will watch that guest for signs of suspicious behavior. If the guest exhibits a repeated pattern of suspicious behavior, the inside guard will kick them out. In other words, an IPS is a second layer of defense that monitors traffic that makes it past the firewall, looking for signs of anomalous behavior.

The other major difference is that IPSs are only concerned with managing incoming traffic, whereas firewalls apply to both incoming and outgoing traffic.



**Note:** An intrusion detection system (IDS) is the same as an IPS except it does not actively block traffic, but only alerts personnel.



An IPS appliance monitoring traffic on a LAN segment.

## DENYHOSTS AND FAIL2BAN

There are many IPS solutions available. Two common third-party solutions are DenyHosts and Fail2ban, both of which examine log files for anomalies.

DenyHosts primarily protects SSH servers from brute force password cracking attacks. In such attacks, an attacker will repeatedly attempt to log in to the SSH server using credentials that increment each time. DenyHosts monitors the authentication log to look for failed login entries. It will take the source IP address and number of failed attempts into consideration. If enough failed attempts from the same source meet the threshold you've configured (or the default), DenyHosts will block that source. A major limitation of DenyHosts is that it only works with IPv4 traffic and not IPv6.

Fail2ban also prevents brute force attacks, but unlike DenyHosts, it does not focus on any one service. Instead, it can monitor log files that pertain to any system service with an authentication component. Fail2ban leverages Netfilter and **iptables** to actually perform blocking actions, and can even be used to update your firewall rules. Fail2ban supports both IPv4 and IPv6.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

## CONFIGURATION

The primary configuration file for DenyHosts is the `/etc/denyhosts.conf` file. There are various settings you can adjust in this file. Some examples include:

- **ADMIN\_EMAIL**—Define what email address to send alerts to.

- **BLOCK\_SERVICE**—Define what services will be blocked from access by unauthorized users.
- **DENY\_THRESHOLD\_VALID**—Defines how many times a user can attempt to log in to an existing account before being blocked.

The primary configuration file for Fail2ban is the **/etc/fail2ban/jail.conf** file. However, if you plan on configuring Fail2ban, it is best to copy this file to **/etc/fail2ban/jail.local** or make a custom .conf file within the **/etc/fail2ban/jail.d/** directory. The following are some example settings:

- **bantime**—Defines how long a host is blocked from accessing a resource.
- **maxretry**—Defines the number of times a host can fail to authenticate before being blocked.
- **ignoreip**—Defines a whitelist of accepted hosts.

# Activity 12-7

## Discussing Firewalls

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. **What type of firewall examines network packets in isolation, rather than in the context of other packets in a transmission?**
  
2. **Several of your users want to install and use Linux at home but are confused by firewall rules and managing a firewall for themselves. What tool can you recommend to help them configure their firewalls?**
  
3. **True or false? The Uncomplicated Firewall (UFW) originated with Ubuntu but can be downloaded and installed on other distributions.**
  
4. **Your parent company purchased another smaller company that has six Linux systems in its server room, all of which are now your responsibility to manage. They all run the firewalld service. Which command can you use to manage the firewalld service and its associated zones?**
  

---

5. **True or false? IP forwarding is the Linux kernel implementation of network routing functionality.**

# Activity 12-8

## Configuring a Firewall

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You will work with a partner in this activity. Earlier, you compiled and installed the Nmap utility.

### SCENARIO

Your fellow network administrators have designed a DMZ in which you'll need to place several public-facing Linux systems. For the most part, these systems function as web servers, and need to allow users to connect using the HTTP and HTTPS protocols. The servers are also running a custom app that the development team has programmed to accept connections on port 7743.

So, your job is to tighten the network security of the servers without denying access to the necessary services. You'll do this by configuring the **firewalld** service, the default and preferred firewall service on CentOS 7.

1. Test the firewall to see if it's blocking a well-known port.

- a) Enter **systemctl status firewalld**
- b) Confirm that the service is currently active and running.



*Note: If it isn't, enter sudo systemctl start firewalld*

- c) Enter **sudo /usr/local/bin/nmap -sV <partner IP address or hostname> -p 80**
- d) Verify that port 80 is listed as "filtered", meaning that a firewall is likely blocking traffic on this port.

2. Get more information about a specific zone.

- a) Enter **firewall-cmd --get-zones** to get a list of all default zones.
- b) Enter **sudo firewall-cmd --zone=dmz --list-all**
- c) Verify that you see several details about this zone, including:
  - Its target. The target defines the behavior for incoming connections (i.e., accept, reject, or drop traffic). The **default** setting is to reject traffic not matching any rules.
  - What network interfaces this zone is currently applied to.
  - What services, ports, and protocols are applied to this zone. In this case, SSH is the only service that will accept incoming connections on interfaces that use this zone.
  - Additional rules for port forwarding, blocking ICMP (ping), and more.
- d) Enter **sudo firewall-cmd --get-active-zones**
- e) Verify that, currently, your main Ethernet interface is using the **public** zone.

3. Set the **dmz** zone for your primary network interface.

- a) Enter **ip a** and note the device name of your primary Ethernet interface. This is the interface that you configured in the networking lesson.
  - b) Enter **Sudo firewall-cmd --zone=dmz --change-interface=<device ID> --permanent**
  - c) Verify that the change was a success.
- 4.** Add services and a custom port to the zone.
- a) Enter the following:  
**sudo firewall-cmd --zone=dmz --add-service=http --permanent**
  - b) Enter the following:  
**sudo firewall-cmd --zone=dmz --add-service=https --permanent**
- You've just added HTTP and HTTPS as services that will apply to the **dmz** zone. In other words, traffic bound for these services will be allowed.
- c) Enter the following:  
**sudo firewall-cmd --zone=dmz --add-port=7743/tcp --permanent**
- This adds a specific port instead of a defined service to the zone.
- 5.** Verify that your configurations were applied to the zone.
- a) Enter **Sudo firewall-cmd --reload**
  - b) Enter **Sudo firewall-cmd --zone=dmz --list-all**
  - c) Verify the following about this zone:
    - It is being used by your main network interface.
    - The HTTP and HTTPS services are active in the zone.
    - TCP port 7743 is also active in the zone.

```
dmz (active)
 target: default
 icmp-block-inversion: no
 interfaces: enp3s0
 sources:
 services: ssh http https
 ports: 7743/tcp
 protocols:
 masquerade: no
 forward-ports:
```

- 6.** Test the firewall to see if your configurations are working as intended.
- a) Wait for your partner to complete the previous steps.
  - b) Enter **Sudo /usr/local/bin/nmap -sV <partner IP address or hostname> -p 80**
  - c) Verify that the port is now open, indicating that the firewall is allowing traffic on this port.
  - d) Use the same **nmap** command to scan for port 21 (FTP).

- e) Verify that it is filtered, indicating that the firewall is blocking traffic on this port as expected.
- f) Enter `ssh student##@<partner's IP address or hostname>`



*Caution: Remember, use your partner's student number.*

- g) Verify that you are able to make a connection, indicating that SSH traffic is allowed.
- h) Enter `exit` to close the SSH session.

**7. What is the Linux kernel framework that most firewall services rely on to some degree?**

**8. In an iptables table, what is the function of a chain, and how do chains interact with one another?**

**9. True or false? IP forwarding is most useful on systems with only one network interface.**

True

False

**10. What are the differences between a firewall and an intrusion prevention system (IPS)?**

---

# Topic E

## Implement Logging Services



### EXAM OBJECTIVES COVERED

3.4 Given a scenario, implement logging services.

Security is not just a process of designing a hardened system. Another major element is the analysis of system, software, and user events. By generating and maintaining logs of these events, you'll be able to more easily identify malicious behavior or misconfigurations that increase the risk of compromise.

### SYSTEM LOGS

System logs are records of system activities and events that are tracked and maintained by the **syslogd** daemon. System logs use the **syslog** standard, which facilitates a centralized logging server that can receive and process syslog data from many systems across a network. This is called remote logging. The syslog standard also supports local logging, where logs are stored on the same system that generated them.

System logs are recorded in simple text files that you can examine and process like any other text. Entries in a syslog system typically include the date and time of the event, the process name and ID that sent the message, and the message itself. The syslog format may also prioritize messages by facility and severity. Facility codes indicate what system was affected, such as "kern" for kernel and "mail" for the mailing system. Severity codes indicate what level of impact the event might have, from 0 (most critical) to 7 (least critical).

### LOG FILE LOCATIONS

In most Linux distributions, system logs are located in the **/var/log/** directory. Inside this directory are the logs themselves, and each file corresponds to a service, application, or feature. The following table lists some of the most common log files.

| Log File                      | Contains                                                                                                 |
|-------------------------------|----------------------------------------------------------------------------------------------------------|
| <b>/var/log/syslog</b>        | All types of system events except for authentication messages. Primarily used by Debian-based distros.   |
| <b>/var/log/messages</b>      | General non-critical system events. Primarily used by RHEL and CentOS.                                   |
| <b>/var/log/auth.log</b>      | Authentication messages (e.g., login successes and failures). Primarily used by Debian-based distros.    |
| <b>/var/log/secure</b>        | Authentication messages. Primarily used by RHEL and CentOS.                                              |
| <b>/var/log/kern.log</b>      | Kernel messages (e.g., <b>dmesg</b> output).                                                             |
| <b>/var/log/[application]</b> | Messages from miscellaneous applications (e.g., <b>cron</b> , <b>firewalld</b> , <b>maillog</b> , etc.). |

## LOG ROTATION

**Log rotation** is the practice of creating new versions of a log file to maintain a minimum log file size. The **logrotate** utility is used to perform automatic rotation of logs. When executed, **logrotate** adds a .1 to the end of the file name of the current version of the log files. Previously rotated files are suffixed with .2, .3, and so on. The utility can also be configured to append the date of the log to the file name rather than a decimal number.

Using automatic rotation, all versions of a log file will be maintained on the system unless otherwise specified. Log files can be rotated on a daily, weekly, monthly, or yearly basis. They can also be automatically rotated if they reach a certain size threshold. Log rotation makes it easier to process and analyze logs, as a log file with too many entries can become unwieldy. Likewise, if the log rotation service is configured to purge older log files, it can save on storage space.

## CONFIGURATION

Log rotation behavior can be configured in the `/etc/logrotate.d/` directory, where each relevant service has its own configuration file. The following is an example configuration file for a service called `myservice`:

```
/var/log/myservice.log {
 size 1k
 create 700 user group
 dateext
 rotate 10
}
```

The first line defines where the log should be output. The **size** directive indicates that the log should rotate when it reaches 1,000 bytes in size. The **create** directive rotates the log file by creating a new one with the specified permissions, user, and group. The **dateext** directive appends the date to the rotated log. Finally, **rotate** specifies that only the 10 most recent log files should be kept.

## THE rsyslog SERVICE

The **syslogd** service is the original syslog service on Linux. The **rsyslogd** service makes several improvements, including support for:

- TCP instead of UDP as the transport protocol, increasing the reliability of transmitted data.
- Data encryption using SSL/TLS.
- Outputting data to various database technologies like MySQL.
- Buffering data on local systems when the remote receiver is not ready to accept it.
- Filtering data based on content.

In addition, **rsyslogd** maintains the same basic configuration format of its predecessor, so it is backwards compatible in that sense.



**Note:** Many modern distributions use `rsyslogd` instead of `syslogd` as their primary logging service.

## THE /etc/rsyslog.conf FILE

The `/etc/rsyslog.conf` file is the configuration file for the **rsyslogd** service. This file determines how to handle syslog messages through a variety of rules that you can modify as needed.

The file takes a two-column format. The first column lists message facilities and/or severities. Severities are defined in word format rather than as numbers 0–7. The second column defines what actions should be taken for messages that correspond to the facility and/or severity. Actions include which file to write the message to; which users to print the message to if they are logged in to a terminal; and which remote hosts to forward the message to.

```
root@server01:~#
#####
Log all kernel messages to the console.
Logging much else clutters up the screen.
#kern.* /dev/console

Log anything (except mail) of level info or higher.
Don't log private authentication messages!
*.notice;mail.none;authpriv.none;cron.none /var/log/messages

```

**Format:**  
facility.severity

**Action (log to)**

The `/etc/rsyslog.conf` file.



**Note:** The `syslogd` equivalent is the `/etc/syslog.conf` file.

## THE syslog-ng SERVICE

The `syslog-ng` service is another replacement for the older `syslogd` service. Although it offers similar functionality to `rsyslogd`, `syslog-ng` has its own syntax.

## THIRD-PARTY AGENTS

Although centralized administration capabilities grant you greater control over logging in your network, the syslog standard is not universally supported on all platforms. Windows, for example, uses the proprietary Windows Event Log format to record system messages. In order to facilitate integration between syslog and non-syslog platforms, you need to use third-party agents. An **agent** is a software program that acts on behalf of some other program or service.

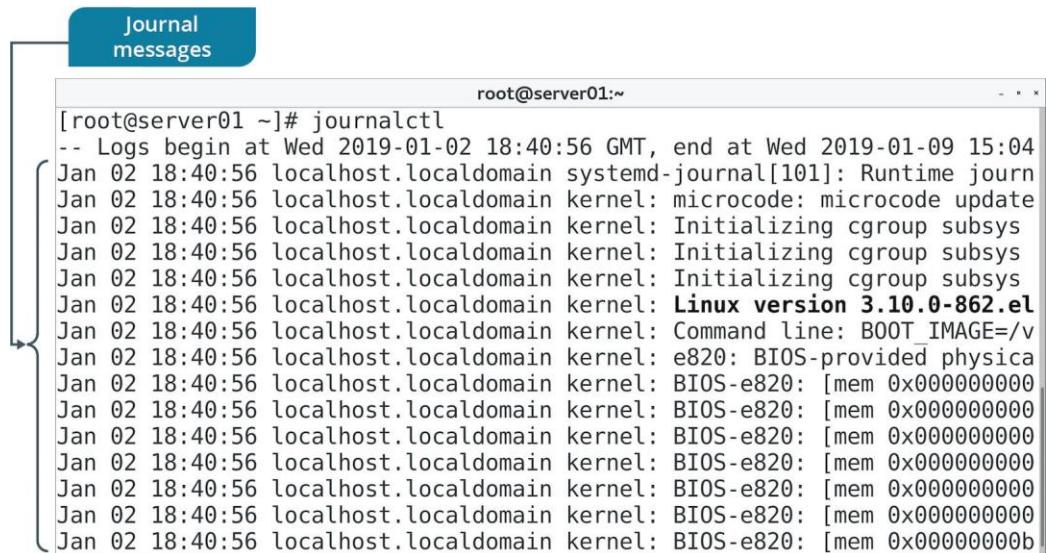
You install a syslog agent on the platform that doesn't normally support the standard, like Windows. Which agent you install will depend on the platform, as well as the type of syslog service you're targeting. For example, `rsyslog` and `syslog-ng` both require their own agent software. Once the agent is installed, you'll be able to configure it to capture messages in a syslog format and send those messages on to your centralized syslog server.

## THE journalctl COMMAND

The `journalctl` command enables you to view and query log files created by the journal component of the systemd suite. Log information is collected and stored via the `systemd journald` service. You can use `journalctl` to print the entire journal log, or you can issue various options with the command to filter the log in a

variety of ways, such as matching a service name or only printing messages matching the specified severity level.

The **journald** service is often used in conjunction with a traditional syslog daemon such as **syslogd** or **rsyslogd**. The settings for **journald** are configured in the **/etc/systemd/journald.conf** file.



```
[root@server01 ~]# journalctl
-- Logs begin at Wed 2019-01-02 18:40:56 GMT, end at Wed 2019-01-09 15:04
Jan 02 18:40:56 localhost.localdomain systemd-journal[101]: Runtime jour
Jan 02 18:40:56 localhost.localdomain kernel: microcode: microcode update
Jan 02 18:40:56 localhost.localdomain kernel: Initializing cgroup subsys
Jan 02 18:40:56 localhost.localdomain kernel: Initializing cgroup subsys
Jan 02 18:40:56 localhost.localdomain kernel: Initializing cgroup subsys
Jan 02 18:40:56 localhost.localdomain kernel: Linux version 3.10.0-862.el
Jan 02 18:40:56 localhost.localdomain kernel: Command line: BOOT_IMAGE=/v
Jan 02 18:40:56 localhost.localdomain kernel: e820: BIOS-provided physica
Jan 02 18:40:56 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000
Jan 02 18:40:56 localhost.localdomain kernel: BIOS-e820: [mem 0x0000000000b
```

*Viewing the systemd journal.*

## SYNTAX

The syntax of the **journalctl** command is **journalctl [options] [matches]**

## journalctl COMMAND OPTIONS

The **journalctl** utility provides a number of options for querying **journald** log data. Some of the frequently used options are listed in the following table.

| Option                      | Used To                                                                                                                         |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <b>-n {number of lines}</b> | Specify the number of lines of journal logs to display.                                                                         |
| <b>-o {output format}</b>   | Specify the format of the output. For example: <b>short</b> , <b>verbose</b> , or <b>export</b> .                               |
| <b>-f</b>                   | Display the most recent journal entries, and continuously update the display with new entries as they are added to the journal. |
| <b>-p</b>                   | Filter journal log output by severity ( <b>alert</b> , <b>err</b> , <b>warning</b> , <b>notice</b> , <b>info</b> , etc.).       |
| <b>-u</b>                   | Filter journal log output by the specified unit, such as the name of a service.                                                 |
| <b>-b [boot ID]</b>         | Show log message from the current boot only, or the boot ID specified.                                                          |

## THE /var/log/journal/ DIRECTORY

In its default configuration, the systemd journal only stores logs in memory, and logs are cleared on each system reboot. You can have the **journald** logs persist after a

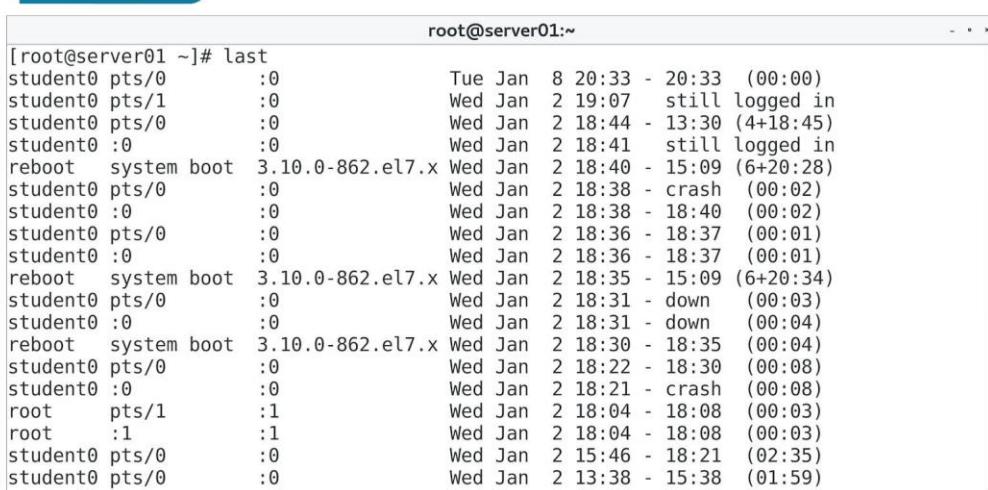
reboot by creating the `/var/log/journal/` directory. The `Systemd` service is configured to automatically maintain logs in this directory if it exists.

## THE `last` COMMAND

The `last` command displays the running history of user login and logout events, along with the actual time and date. It also has various options that enable you to filter the results, such as filtering by users who have logged in through a specific terminal.

For example, `last 1` will display the details of users who logged in using the first terminal (`tty1`). The `last` command retrieves information from the `/var/log/wtmp` file.

To pull this same information for only failed login events, you can use the `lastb` command. This command retrieves information from the `/var/log/btmp` file.



```
[root@server01 ~]# last
student0 pts/0 :0 Tue Jan 8 20:33 - 20:33 (00:00)
student0 pts/1 :0 Wed Jan 2 19:07 still logged in
student0 pts/0 :0 Wed Jan 2 18:44 - 13:30 (4+18:45)
student0 :0 :0 Wed Jan 2 18:41 still logged in
reboot system boot 3.10.0-862.el7.x Wed Jan 2 18:40 - 15:09 (6+20:28)
student0 pts/0 :0 Wed Jan 2 18:38 - crash (00:02)
student0 :0 :0 Wed Jan 2 18:38 - 18:40 (00:02)
student0 pts/0 :0 Wed Jan 2 18:36 - 18:37 (00:01)
student0 :0 :0 Wed Jan 2 18:36 - 18:37 (00:01)
reboot system boot 3.10.0-862.el7.x Wed Jan 2 18:35 - 15:09 (6+20:34)
student0 pts/0 :0 Wed Jan 2 18:31 - down (00:03)
student0 :0 :0 Wed Jan 2 18:31 - down (00:04)
reboot system boot 3.10.0-862.el7.x Wed Jan 2 18:30 - 18:35 (00:04)
student0 pts/0 :0 Wed Jan 2 18:22 - 18:30 (00:08)
student0 :0 :0 Wed Jan 2 18:21 - crash (00:08)
root pts/1 :1 Wed Jan 2 18:04 - 18:08 (00:03)
root :1 :1 Wed Jan 2 18:04 - 18:08 (00:03)
student0 pts/0 :0 Wed Jan 2 15:46 - 18:21 (02:35)
student0 pts/0 :0 Wed Jan 2 13:38 - 15:38 (01:59)
```

*Listing recent login and logout events.*

## SYNTA

### X

The syntax of the `last` command is `last [options]`

## THE `lastlog` COMMAND

The `lastlog` command is similar to the `last` command, but instead of listing the most recent login events, it lists all users and the last time they logged in. This command retrieves information from the `/var/log/lastlog` file.

 **Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 12-9

## Discussing Logging Services

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. You have set up a training session for new Linux system administrators and you are discussing log files, their format, their location, and their contents. Generally speaking, where are most Linux system logs located?
  
2. You are giving a presentation to the IT and Security teams to convince them that you should adopt a new corporate standard for syslog services on your Linux systems. Your argument is that rsyslogd has several advantages or improvements over the standard syslogd service. What are some of those improvements?
  
3. Some files were deleted from a shared directory after regular business hours last night and the other users want to know who was responsible for the deletions. To find out who was on the system during the suspected times, you use the last command. From which log file does the last command read its data?
  
4. You suspect that some users have not logged into the file server to update their passwords since the new file server went into production. Which command lists all users and the last time they logged in?
  
5. You need to set up a log rotation schedule that's relatively maintenance-free. In other words, you don't want to have to purge logs from 30 servers on a regular basis—you want it to happen automatically. Which utility provides this service on a schedule that you determine?

# Activity 12-10

## Configuring rsyslog for Local and Remote Logging

### BEFORE YOU BEGIN

You are logged in to GUI as your student account. You will work with a partner in this activity.

### SCENARIO

You want to ensure that your system is logging only messages that are useful to you and other analysts. So, you'll configure the local logging behavior to be more fine-tuned and less noisy. In addition, each system should be sending its authentication logs to a remote, centralized server for easy analysis and storage. So, you'll configure rsyslog to send these messages to a remote host over the network.

1. Examine the default rules in the rsyslog configuration.

- a) Using **SUDO**, open the **/etc/rsyslog.conf** file in the text editor of your choice.
- b) Scroll down to the **RULES** section.
- c) Verify that, in the left column, each line lists one or more severities and/or facilities.

```
RULES

Log all kernel messages to the console.
Logging much else clutters up the screen.
#kern.* /dev/console

Log anything (except mail) of level info or higher.
Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none /var/log/messages

The authpriv file has restricted access.
authpriv.* /var/log/secure

Log all the mail messages in one place.
mail.* -/var/log/maillog

Log cron stuff
cron.* /var/log/cron
```

For example, the **authpriv.\*** facility refers to private authentication messages, such as login and logout events. The asterisk (\*) indicates that all severities should be logged.

- d) Verify that each line has a corresponding action in the right column.

For example, the **authpriv.\*** facility will log its messages in the **/var/log/secure** file.

2. Filter the **messages** log to reduce the number of events it records.

- a) Locate the line that logs events to the **/var/log/messages** file.

This line tells **rsyslogd** to log all events that are of **info** level (6) severity and above (lower number is more severe). It makes exceptions for mail messages, private authentication messages, and cron messages (scheduler).

- b) On this line, replace the **\*.info** text with **\*.notice**

The **notice** severity (level 5) logs normal but significant conditions. Now, the **messages** log will not record informational messages (level 6), but start with level 5 messages.

3. Configure your server to receive remote rsyslog messages over TCP.

- a) Scroll up to the **MODULES** section.

- b) Find the line that says **# Provides TCP syslog reception**

- c) Change the two lines below so that they read as follows:

```
$ModLoad imtcp
$InputTCPServerRun 601
```



**Note:** Ensure there are no # symbols at the beginning of these lines.

4. Configure the client to send rsyslog traffic to your partner's server.

- a) Scroll to the bottom of the file, and on a new line, type the following:

```
authpriv.* @@<IP address>:601
```

Replace **<IP address>** with your partner's IP address. For example:

```
authpriv.* @@10.50.1.101:601
```



**Note:** Ensure there are no leading # symbols on this line.

The @ @ symbols indicate a TCP connection, whereas a single @ indicates UDP.

- b) Save and close the file.
- c) Enter **sudo systemctl restart rsyslog**

5. Add an allow rule to the firewall for the rsyslog traffic.

- a) Enter the following:

```
sudo firewall-cmd --zone=dmz --add-port=601/tcp --permanent
```

- b) Enter **sudo systemctl restart firewalld**

6. Generate an authentication failure message and confirm it was sent to your partner's server.

- a) Enter **SU - ariley**

- b) Provide an incorrect password and verify that you failed to log in as this user.

- c) Wait for your partner to finish performing all of the previous steps.

- d) Enter **sudo tail /var/log/secure | grep ariley**

- e) Verify that you see an authentication failure message that was sent from your partner.

```
Dec 14 16:33:26 server02 unix_chkpwd[7929]: password check failed for user (ariley)
Dec 14 16:33:26 server02 su: pam_unix(su-l:auth): authentication failure; logname=stud
ent01 uid=1001 euid=0 tty=pts/1 ruser=student02 rhost= user=ariley
```

7. Turn off remote logging for classroom purposes.

- a) Using **SUDO**, open the **/etc/rsyslog.conf** file in the text editor of your choice.
  - b) Scroll to the bottom of the file and remove the entire **authpriv** line.
  - c) Save and close the file.
-

# Activity 12-11

## Examining Log Files

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

### SCENARIO

Up until now, you've examined a few logs using standard tools like **cat** and **less**, and also filtered those logs using a tool like **grep**. However, you can also use **journalctl** to more efficiently shape those log messages in order to extract the specific information you're looking for. You'll also use **last** and its associated commands to get an overview of login events, such as when a user last logged in.

1. Use **journalctl** to retrieve and filter messages.
  - a) Enter **sudo journalctl** and verify that you can page through many lines.
  - b) Press **q** to quit.
  - c) Enter **sudo journalctl -p notice** and verify that the log was filtered. Only messages matching severity level 5 appear.
  - d) Press **q** to quit.
  - e) Enter **sudo journalctl -p notice | grep kernel**
  - f) Verify that kernel messages are displayed.  
These messages are similar to those in the **/var/log/messages** file.
  - g) Enter **sudo journalctl -f** to retrieve the most recent entries.
  - h) Press **Ctrl+C** to terminate the process.
  - i) Enter **sudo journalctl --since "2 hours ago" --until "30 minutes ago"**
  - j) Page through the results and verify that the first entry was two hours ago and the last entry was 30 minutes ago.
  - k) Press **q** to quit.
  - l) Enter **sudo journalctl -u httpd.service**
  - m) Verify that the log is filtered by Apache service messages.
  - n) Press **q** to quit.
2. Use **last** and related commands to examine account login events.
  - a) Enter **last**
  - b) Verify that you can see the login and logout events for various users.
  - c) Enter **sudo lastb**
  - d) Verify that you can see a list of user accounts and the times that an authentication attempt with that account failed.
  - e) Enter **lastlog**
  - f) Verify that you can see the last time that each user logged in.

# Topic F

## Back Up, Restore, and Verify Data



### EXAM OBJECTIVES COVERED

- 2.3 Given a scenario, create, modify, and redirect files.
- 2.6 Given a scenario, back up, restore, and compress files.

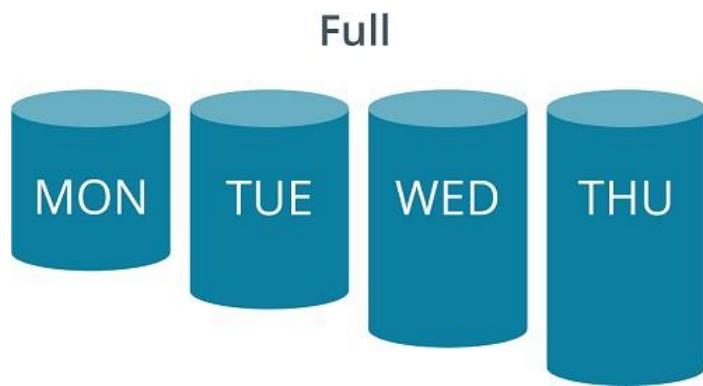
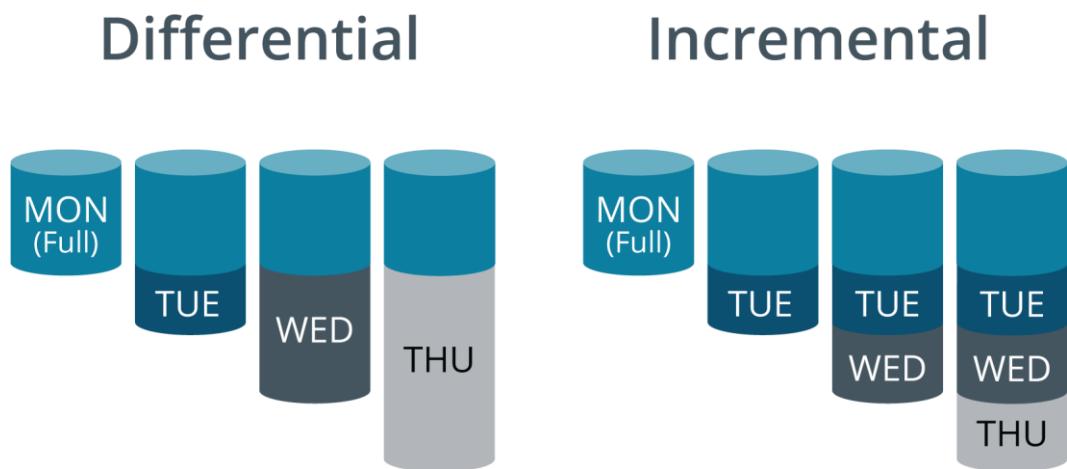
Another major element of cybersecurity is the backup process. Backing up data is crucial in any organization that cannot afford to lose data or suffer damage to that data's integrity. So, in this topic, you'll back up sensitive data, test your ability to restore it at a later time, and then verify that the data was not tampered with.

### BACKUP TYPES

A **backup** is a copy of data that exists in another logical or physical location than the original data. Backups facilitate the recovery process in case of data loss. The process of recovering data from a backup varies depending on the backup types that were included in the original backup plan.

There are three main types of backups.

| Backup Type                | Description                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Full backup</b>         | All selected files, regardless of prior state, are backed up. Numerous full backups can consume a great deal of storage space, and the backup process can be slow. However, full backups are fast and reliable when it comes to recovering lost data.                                                                                                                                                                          |
| <b>Differential backup</b> | All selected files that have changed since the last full backup are backed up. When differential backups are used, you must restore the last full backup plus the most recent differential backup. Differential backups require less storage space and backup time than full backups, but are slower to recover.                                                                                                               |
| <b>Incremental backup</b>  | All selected files that have changed since the last full or incremental backup (whichever was most recent) are backed up. When incremental backups are used, you must restore the last full backup plus all subsequent incremental backups. An incremental backup typically takes less time to perform than a differential backup because it includes less data, but it is also slower when it comes time to recover the data. |

*A full backup.**Differential vs. incremental backups.*

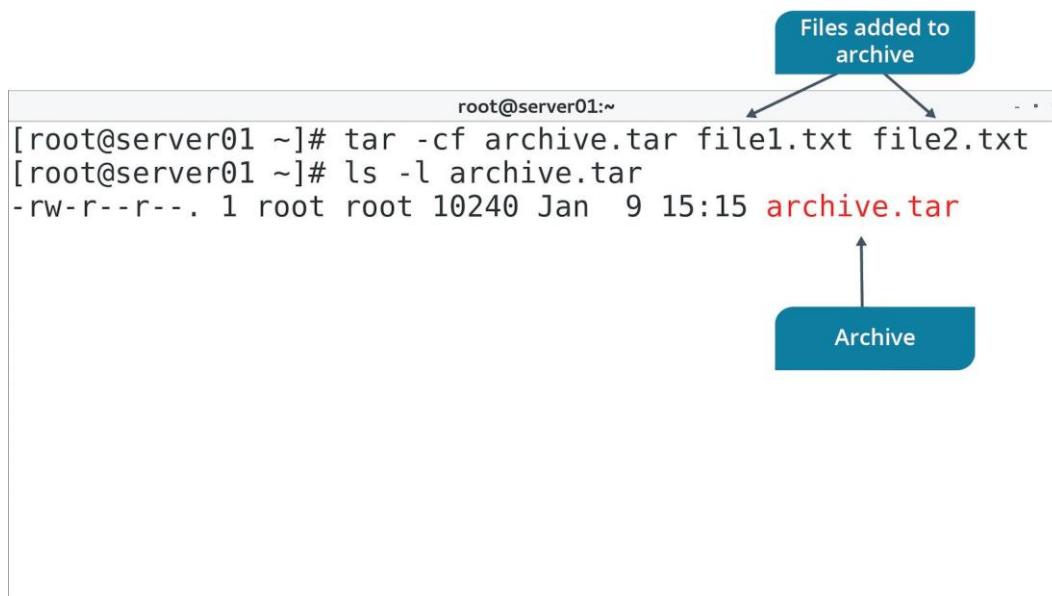
## BACKUP STORAGE METHODS

Full, differential, and incremental all describe the frequency of data backups. But there are also different ways to go about storing backup data.

| Backup Storage Method | Description                                                                                                                                                                                                                                                                                                                  |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Snapshot              | Snapshots record the state of a storage drive at a certain point in time and usually exist on the same drive. They are "checkpoints" that you can restore the drive to rather than true copies of data that exist elsewhere.                                                                                                 |
| Image                 | You can back up data as individual files or as collections of files, but you can also create one-to-one copies of entire systems. Image-based backups save the state of an operating system in an image file format like ISO. You can use this image to restore a system to the state it was in when the image was captured. |
| Clone                 | Cloning is the process of copying all of the contents of a storage drive to another storage medium. Technically, an image backup is a clone of a drive. However, cloning operations often go one step further by using the image file to reconstruct the original drive on a second drive.                                   |

## THE tar COMMAND

The **tar** command enables you to create archives of data. It's commonly used to create an archive file from a directory that contains the data you want to back up. You can also use the command on previously created archives to extract files, store additional files, update files, and list files that were already stored. File archives made with **tar** frequently have the .tar file extension. The **tar** command can also direct its output to available devices, files, or other programs using pipes.



*Creating an archive from multiple files.*

### SYNTA

#### X

The syntax of the **tar** command is **tar [options] {file names}**

### RESTORING FILES WITH THE tar COMMAND

The command **tar -xvf** will restore the entire contents of the source file or directory structure. To restore a portion of a tar file, use the path and name of the file you wish to extract. You must use the exact path and name that was used when you created the tar file. You can also make restores interactive by using the command **tar -wxvf [destination] [source]**

### THE dar COMMAND

The **dar** ("disk archiver") command is intended to replace **tar** by offering more backup and archiving functionality. It is especially useful at creating full, differential, and incremental backups. The following command creates a full backup of the **mydata** directory and outputs a backup file named **full.bak**:

**dar -R mydata -c full.bak**

To create a differential backup (**diff1.bak**), you can reference the full backup using the **-A** option:

**dar -R mydata -c diff1.bak -A full.bak**

You can then create more differential backups as needed by referencing the full backup with the **-A** option. However, to perform incremental backups instead, you need to reference the previous incremental backup, like so:

```
dar -R mydata -c incr1.bak -A full.bak
dar -R mydata -c incr2.bak -A incr1.bak
```

The **-X** (extract) option is used to recover a backup. If you performed differential backups, you need to first extract the full backup, then the latest differential backup:

```
dar -x full.bak
dar -x diff1.bak -w
```

The **-W** option automatically overwrites changes to files; otherwise, you will be prompted to confirm.

To recover an incremental backup, you need to first extract the full backup, then each incremental backup, in order:

```
dar -x full.bak
dar -x incr1.bak -w
dar -x incr2.bak -w
```

## THE cpio COMMAND

The **cpio** command copies files to and from archives. The **cpio** command has three operating modes.

| Operating Mode | Command        | Used To                                                                                                                                                                 |
|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Copy-out       | <b>cpio -o</b> | Copy files into an archive. It reads the standard input to obtain a list of file names and then copies those files to the standard output.                              |
| Copy-in        | <b>cpio -i</b> | Copy files from an archive. It extracts files from the standard input. This option is used in data recovery.                                                            |
| Copy-pass      | <b>cpio -p</b> | Copy files from one directory tree to another. It reads the standard input to obtain the list of file names that are created and copied into the destination directory. |

## SYNTAX

The syntax of the **cpio** command depends on its mode. In all modes, the command reads from standard input. The following copy-out example archives all contents of a directory by piping **ls** to **cpio** and sending the archive output to **dir\_arch**:

```
ls | cpio -o > dir_arch
```

In copy-in mode, you can extract an archive as follows:

```
cpio -i < dir_arch
```

In copy-pass mode, you can pipe **find** to **cpio** to copy one directory tree to another:

```
find . -depth -print | cpio -p new_dir
```

## THE dd COMMAND

The dd command copies and converts files to enable them to be transferred from one type of media to another. The dd command has various operands, or actions, to perform.

| Operand               | Used To                                                                                                                                                                                                                                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>if={file name}</b> | Specify the file from which data will be read.                                                                                                                                                                                                                                                                     |
| <b>of={file name}</b> | Specify the file to which data will be written.                                                                                                                                                                                                                                                                    |
| <b>bs={bytes}</b>     | Specify the total block size to read and write, in bytes.<br>Bytes can also be formatted in a more human-friendly way, such as <b>50M</b> to specify 50 megabytes and <b>10G</b> to specify 10 gigabytes.                                                                                                          |
| <b>count={count}</b>  | Specify the number of blocks to be written to the output file from the input file.                                                                                                                                                                                                                                 |
| <b>status={level}</b> | Specify the level of information to print to standard error: <ul style="list-style-type: none"> <li>• <b>none</b> to suppress everything except error messages.</li> <li>• <b>noxfer</b> to suppress total transfer statistics.</li> <li>• <b>progress</b> to display transfer statistics periodically.</li> </ul> |



**Note:** A selected input file is copied to a selected output file. If no files are selected, the standard input and the standard output are used.

## SYNTAX

The syntax of the dd command is **dd [options] [operands]**

## USING dd FOR BACKUPS

You can use dd to perform a full backup of a storage partition. The following example copies data from /dev/sda1 to /dev/sdb2:

```
dd if=/dev/sda of=/dev/sdb
```

Using dd, you can also create an image of a drive and then clone a second drive with it:

```
dd if=/dev/sda of=drive_image.iso
dd if=drive_image.iso of=/dev/sdb
```

## THE mirrorvg COMMAND

The mirrorvg command creates copies, or mirrors, of all logical volumes in a specified logical volume group. By default, the command will create the mirrors on the same drives that are associated with the volume group. You can also specify other drives to mirror the volumes to, if desired. The -C option can be used to create two or three copies of a logical volume, rather than the default of just a single copy.

## SYNTAX

The syntax of the mirrorvg command is **mirrorvg [options] {volume group}**

## OTHER WAYS TO MIRROR LOGICAL VOLUMES

Other than using **mirrorvg** to mirror all volumes in a group, you can also use the **mklvcopy** command to mirror individual logical volumes in a volume group. You can also use the **-m#** option with **lvcreate** to create one or more mirrors of a logical volume. For example, the following command creates one 10 GB mirror called **mirrorlv** that copies from the **volgr** volume group:

```
lvcreate -L 10G -m1 -n mirrorlv volgr
```

## OFF-SITE BACKUP

An **off-site backup** is a physical location outside of the main site that stores copies of data. Off-site backups are a component of disaster recovery processes, as they are often necessary to reconstruct data in the event of a major disruption to the main site.

There are several ways to copy data from the main site to the backup site. Rather than physically move backup storage devices from one location to the other, it's usually more convenient to connect both sites by a network, such as in a VPN, and transfer data over that network.

## DATA TRANSFER TOOLS

The following data transfer tools are useful in facilitating the off-site backup process.

| Data Transfer Tool | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>scp</b>         | This tool is used to copy data to or from a remote host over SSH. Because it uses SSH, data you send to an off-site backup will be encrypted in transit, protecting its confidentiality. Like SSH, <b>scp</b> uses TCP port 22 by default. The following is an example of copying a file to a remote host:<br><br><b>scp file.txt user@host:/home/dir</b>                                                                                                                                                                                                                                                                                                             |
| <b>sftp</b>        | This command is the implementation of the Secure File Transport Protocol (SFTP). SFTP uses SSH tunnel as a transportation mechanism to encrypt data. Whereas <b>scp</b> is used purely for transferring files, <b>sftp</b> can transfer files <i>and</i> manage files and directories. So, you can list, create, and remove directories on the remote system. The <b>sftp</b> command also supports resuming file transfers, whereas <b>SCP</b> does not.<br><br>Just like with the standard <b>ftp</b> command, you can use <b>sftp</b> interactively or non-interactively. For example, to retrieve a file non-interactively:<br><br><b>sftp user@host:file.txt</b> |

| Data Transfer Tool | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rsync              | <p>This tool is used to copy files locally and to remote systems. Its real power lies in its efficient use of network bandwidth; instead of copying all files, it only copies differences between files. So, if you use <b>rsync</b> on all files in a directory, it will check the destination directory to see if those exact files already exist. Only files that aren't already in the destination will be copied.</p> <p>The <b>rsync</b> command can copy files over SSH, or it can use the <b>rsyncd</b> daemon if you set it up on the remote system. The following is an example of synchronizing the files in a local directory to a remote directory over SSH:</p> <pre>rsync -a /home/mydir/ user@host:/home/mydir/</pre> |

 **Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 12-12

## Archiving and Restoring Files

### DATA FILES

All files in:

/opt/linuxplus/securing\_linux\_systems/employee\_data

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have an encrypted / backup/data volume.

### SCENARIO

On a yearly basis, HR has been compiling information about Develetech employees and putting them in a spreadsheet. These spreadsheets contain personal information such as names, addresses, and phone numbers. You need to ensure that past years' reports are stored in backup should they ever need to be retrieved in the future.

Because you won't need to regularly update these files, you decide to place them in a single archive. You also want to test the process of recovering the files from this archive if it's ever necessary.

1. If necessary, unlock the encrypted volume.



*Note: Open the **Files** app, then select + Other Locations, then select the encrypted volume and input **linuxplus** as the passphrase.*

2. Copy the employee data files to your home directory.
  - a) Ensure you are in your home directory.
  - b) Enter the following:  
`cp -r /opt/linuxplus/securing_linux_systems/employee_data  
employee_data`
  - c) Enter **ls -l** and verify that the directory was copied.
3. Archive the employee data files and then copy the archive to the data backup volume.
  - a) Enter **tar -cvf employee\_data.tar employee\_data/\***  
This creates a new archive with the specified name. The asterisk (\*) indicates that all files within the **employee\_data** directory should be added to the archive.
  - b) Enter **ls -l** and verify that the .tar file is present.
  - c) Enter **tar -tf employee\_data.tar** to list the contents of the archive.
  - d) Enter **sudo cp employee\_data.tar /backup/data/employee\_data.tar**

- e) Enter **ls -l /backup/data** and verify that the archive file is now on the data backup volume.
4. Restore all files from the archive, then restore a single file.
- a) Enter **cd /backup/data** to change your current working directory.
  - b) Enter **Sudo tar -xf /backup/data/employee\_data.tar**
  - c) Enter **ls -l employee\_data** and verify that all of the files were extracted to the directory.
  - d) Enter **sudo rm -r employee\_data** to delete the directory.
  - e) Enter **sudo tar -xf employee\_data.tar employee\_data/emp\_2018.csv**
  - f) Enter **ls -l employee\_data** and verify that only the one file was extracted.
  - g) Enter **sudo rm -r employee\_data** to delete the directory.
-

# Activity 12-13

## Synchronizing Files

### DATA FILES

All files in:

/opt/linuxplus/securing\_linux\_systems/prototypes

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have an encrypted / backup/data volume.

### SCENARIO

The Research and Development (R&D) department also has sensitive data that they need backed up: data on product prototypes, including model numbers, pricing, and release dates. Unlike the employee data, these files are likely to be updated regularly. So, you want to make sure the backup copies consistently align with the source copies. You'll use **rsync** to synchronize both copies, ensuring that the backup copies will only need to be updated if the source files have changed.

1. Copy the prototype product files to your home directory.
  - a) Enter **cd ~** to return to your home directory.
  - b) Enter the following:  
**cp -r /opt/linuxplus/securing\_linux\_systems/prototypes prototypes**
  - c) Enter **ls -l** and verify that the directory was copied.
2. Synchronize the prototype files with a directory on the backup volume.
  - a) Enter **sudo rsync -av prototypes /backup/data**
  - b) In the output, verify that each file in the folder was copied, and that the command sent a specific number of bytes.
  - c) Enter **ls -l /backup/data/prototypes** and verify that all files were copied.
3. Make a change to a file and resynchronize with the backup directory.
  - a) Enter **echo -e "\nSW950,749.99,12/5" >> prototypes/swatch.csv** to make a change to one of the files.
  - b) Enter **sudo rsync -av prototypes /backup/data**

- c) In the output, verify that the only file that was sent was the one that you changed.

```
[student01@server01 ~]$ sudo rsync -av prototypes /backup/data
sending incremental file list
prototypes/swatch.csv

sent 377 bytes received 36 bytes 826.00 bytes/sec
total size is 667 speedup is 1.62
```

- d) Enter `cat /backup/data/prototypes/swatch.csv` and verify that your change was added to the backup version of the file.

## COMPRESSION

**Compression** is a procedure in which data is encoded to reduce the amount of bits that are used to represent that data. The compression process can significantly reduce the size of a file or collection of files to make the storage and transfer of data more efficient. Although the file takes up less space, it still contains the requisite information so that only redundant data is removed (lossless compression) or so that only non- critical data is lost (lossy compression).

Compression is commonly used to reduce the storage and transmission burden involved with creating, maintaining, and recovering from backups. Rather than backing up data one-to-one, you can compress that data and then store it.

## THE gzip COMMAND

GNU zip (`gzip`) is a compression utility that reduces the size of selected files. Files compressed with `gzip` frequently have the `.gz` file extension. The `gzip` command has several options. These command options are described in the following table.

| Option          | Used To                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------|
| <code>-d</code> | Reverse file compression (decompression).                                                         |
| <code>-f</code> | Force compression or decompression of a file even if it has multiple links or if the file exists. |
| <code>-n</code> | Omit saving the original file name and timestamp.                                                 |
| <code>-N</code> | Save the original file name and timestamp.                                                        |
| <code>-q</code> | Suppress all warnings.                                                                            |
| <code>-r</code> | Enable directory recursion during compression or decompression.                                   |
| <code>-v</code> | Display the name and percentage reduction of the compressed or decompressed file.                 |
| <code>-t</code> | Perform an integrity check on the compressed file.                                                |

```
root@server01:~#
[root@server01 ~]# ls -l archive.tar
-rw-r--r--. 1 root root 10240 Jan 9 15:20 archive.tar
[root@server01 ~]# gzip archive.tar
[root@server01 ~]# ls -l archive.tar.gz
-rw-r--r--. 1 root root 141 Jan 9 15:20 archive.tar.gz
[root@server01 ~]#
```

*Compressing an archive file with gzip.*

## SYNTAX

The syntax of the **gzip** command is **gzip [options] [file names]**

## THE gunzip COMMAND

The **gunzip** command is equivalent to issuing **gzip -d** at the command-line.

## THE xz COMMAND

The **XZ** command is a data compression utility, similar to **gzip**, that reduces the size of selected files and manages files in the **.xz** file format. The **XZ** command has several options.

| Option    | Used To                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------|
| <b>-d</b> | Decompress a file.                                                                                |
| <b>-f</b> | Force compression or decompression of a file even if it has multiple links or if the file exists. |
| <b>-q</b> | Suppress all warnings.                                                                            |
| <b>-v</b> | Display the name and percentage reduction of the compressed or decompressed file.                 |
| <b>-t</b> | Perform an integrity check on the compressed file.                                                |

```
root@server01 ~]# ls -l archive.tar
-rw-r--r--. 1 root root 10240 Jan 9 15:22 archive.tar
[root@server01 ~]# xz archive.tar
[root@server01 ~]# ls -l archive.tar.xz
-rw-r--r--. 1 root root 180 Jan 9 15:22 archive.tar.xz
[root@server01 ~]#
```

Compressing an archive file with xz.

## SYNTAX

The syntax of the xz command is `xz [options] [file names]`

## THE bzip2 SUITE

The bzip2 command and its related commands manage file compression. Files compressed with bzip2 frequently have the .bz2 file extension. The bzip2-related commands are described in the following table.

| Command       | Used To                                                |
|---------------|--------------------------------------------------------|
| bzip2         | Compress a file.                                       |
| bunzip2       | Decompress a file.                                     |
| bzcat         | Decompress a file to standard output.                  |
| <b>bzdiff</b> | Run the <code>diff</code> command on compressed files. |
| bzip2recover  | Recover data from damaged .bz2 files.                  |
| bzless        | Run the <code>less</code> command on compressed files. |
| bzmore        | Run the <code>more</code> command on compressed files. |



**Note:** Archives made with `tar` are frequently compressed with `gzip` (resulting in the file extension `.tar.gz`) or `bzip2` (resulting in the file extension `.tar.bz2`).

```
root@server01:~#
[root@server01 ~]# ls -l archive.tar
-rw-r--r--. 1 root root 10240 Jan 9 15:23 archive.tar
[root@server01 ~]# bzip2 archive.tar
[root@server01 ~]# ls -l archive.tar.bz2
-rw-r--r--. 1 root root 133 Jan 9 15:23 archive.tar.bz2
[root@server01 ~]#
```

*Compressing an archive file with bzip2.*

## SYNTAX

The syntax of the **bzip2** command is **bzip2 [options] {file names}**

For example, to compress files **file1** and **file2**:

**bzip2 file1 file2**

## THE zip COMMAND

The **zip** command is another compression utility, but unlike **gzip**, **xz**, and **bzip2**, it also features file archiving functionality. In fact, **zip** is a combination of an older compression utility called **compress** and the **tar** archive command. Files compressed with **zip** frequently have the **.zip** file extension. The **zip** command has several options.

| Option | Used To                                         |
|--------|-------------------------------------------------|
| -d     | Delete entries in a .zip archive.               |
| -e     | Encrypt the contents of an archive.             |
| -F     | Fix a corrupted .zip archive.                   |
| -r     | Enable recursion.                               |
| -T     | Perform an integrity check on the archive file. |

## SYNTAX

The syntax of the **zip** command is **zip [options] [file names]**

## WHICH COMPRESSION METHOD SHOULD YOU CHOOSE?

Which compression tool to use will often depend on your own particular needs, but some generalities can be made about each. The most important factors are the speed/ time of compression and decompression and the compression ratio, which is the size of the uncompressed file divided by the size of the compressed file (e.g., a 5 MB uncompressed file that becomes 1 MB when compressed has a ratio of 5:1).

For compression speed, tests tend to show that **gzip** is slightly faster than **bzip2**, and both are significantly faster than **XZ** when the applied compression level increases. For decompression speed, **gzip** tends to be the fastest again, with **XZ** and

bzip2 as second and third fastest, respectively. When it comes to compression ratio, xz tends to perform the best, followed by bzip2, with gzip having the worst ratio.

Ultimately, consider using:

- gzip if you just care about compressing and decompressing files as fast as possible and are less concerned with storage space.
- xz if storage space is at a premium, and time is not as much of a factor.
- bzip2 to strike a balance, and for data that rarely needs to be decompressed.



**Note:** Consider visiting <https://www.rootusers.com/gzip-vs-bzip2-vs-xz-performance-comparison/> for a more detailed testing report.



**Note:** Because gzip uses an older compression algorithm, it does not perform as well as the other three mentioned.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 12-14

## Compressing Files

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have created a .tar archive of employee data files.

### SCENARIO

As you archive more and more data, you realize that the archives take up just as much space as the files they hold. This is a waste of space, especially if you're not working with the archives' contents regularly. So, you'll compress the archives so that they take up significantly less space without losing any data. You'll also try several different compression algorithms and compare their performance.

1. Note the size of the employee data archive on the backup volume.
  - a) Enter `cd /backup/data`
  - b) Enter `ls -lh` and note the size of `employee_data.tar` that you created earlier.
  
2. Compress the archive with gzip
  - a) Enter `sudo bash -c "gzip -cv employee_data.tar > employee_data.tar.gz"`

By default, `gzip` replaces the file with the compressed version. With this command, you are using the `-C` option with output redirection to keep the original .tar file intact.
  - b) Verify that the output indicates that the file was reduced by over 90% of its original size.
  - c) Enter `ls -lh` and confirm that `employee_data.tar.gz` is much smaller in size.
  
3. Decompress the archive.
  - a) Enter `sudo tar -xzf employee_data.tar.gz`
  - b) Enter `ls -l employee_data` and verify that the individual files were extracted.
 

With the `-Z` option, the `tar` command has the ability to use `gzip` to compress and decompress archives. You could use `gzip` to decompress the file and then `tar` to unarchive it, but this is faster.
  - c) Enter `sudo rm -r employee_data` to delete the directory.
  
4. Compress and decompress the archive with xz
  - a) Enter `sudo xz -kv employee_data.tar`

The `-K` option keeps the original file intact.

- b) Verify that the output indicates that the compressed file is less than 5% the size of the original (expressed in decimal).

```
employee_data.tar (1/1)
100 % 8,040 B / 300.0 KiB = 0.026
```

- c) Enter **ls -lh** and verify that the **employee\_data.tar.xz** file is even smaller than the .gz file.

- d) Enter **sudo tar -xJf employee\_data.tar.xz**

The **tar** command can also work with .xz files through the **-J** option. As before, this decompresses and unarchives the .tar archive all in one command.

- e) Enter **ls -l employee\_data** and verify that all of the files are there.

- f) Enter **sudo rm -r employee\_data** to delete the directory.

## 5. Compress and decompress the archive with bzip2

- a) Enter **sudo bzip2 -kv employee\_data.tar**

- b) Verify that the output indicates that the file was reduced by over 90% of its original size.

- c) Enter **ls -lh** and confirm that **employee\_data.tar.bz2** is slightly larger than the .gz equivalent.

- d) Enter **sudo tar -xjf employee\_data.tar.bz2**

The **tar** command can also work with .bz2 files through the **-j** option. As before, this decompresses and unarchives the .tar archive all in one command.

- e) Enter **ls -l employee\_data** and verify that all of the files are there.

- f) Enter **sudo rm -r employee\_data** to delete the directory.
- 

## INTEGRITY CHECKING

**Integrity checking** is the process of verifying that data has not been modified, whether intentionally or unintentionally, in any way. In other words, an integrity check can validate the security goal of integrity. It is good practice to perform integrity checking after you finish compressing and archiving a backup file to confirm that the data has not changed. This will help you avoid storing corrupted and inaccurate archives for future recovery, only to find out too late that the data was not properly backed up.

There are several methods that enable you to check data integrity, each of which may vary based on its security requirements or goals. One of the most common and secure methods of checking data integrity is through the use of hashing. By calculating the hash of a file like a backup archive, you can compare that hash to past values, and if both are the same, you can be reasonably sure the data has not changed in the meantime.

## HASH FUNCTIONS

A **hash function** is an algorithm that performs a hashing operation. There are many different hash functions, each of which may have its own security strengths and weaknesses. The two most common hash functions for checking data integrity on Linux systems are MD5 and SHA.

The **Message Digest 5 (MD5)** algorithm produces a 128-bit message digest. It was created by Ronald Rivest and is now in the public domain. MD5 is no longer considered a strong hash function and should be avoided for sensitive operations like storing passwords; however, it is still used in integrity checking.

The **Secure Hash Algorithm (SHA)** algorithm is modeled after MD5 and is considered the stronger of the two. Common versions of SHA include SHA-1, which produces a 160-bit hash value, while SHA-256, SHA-384, and SHA-512 produce 256-bit, 384-bit, and 512-bit digests, respectively. SHA-1 is being deprecated due to some security weaknesses.

## THE md5sum COMMAND

The **md5sum** command is used to calculate the hash value of a file or standard input using the MD5 hash function. You can also use the **-C** option to specify a file containing MD5 hashes and the file names they apply to; **md5sum** will calculate the hashes of the files listed, and then compare them to the hash values listed. The results will let you know if each file passed, failed, or could not be found.

MD5 hashes are 128-bits in length. Like many other hash values, they are typically represented in hexadecimal format (32 characters for MD5). The following is the hash value of the string "Linux":

```
edc9f0a5a5d57797bf68e37364743831
```

### SYNTAX

The syntax of the **md5sum** command is **md5sum [options] [file name]**

## SHA COMMANDS

There are several different commands that you can use to calculate SHA hash values. These commands are functionally identical to **md5sum**, but use the SHA function with the applicable bit size:

- **sha1sum**
- **sha256sum**
- **sha384sum**
- **sha512sum**

```

Initial hash
[Initial file content]
Change to file
[echo "1" >> file1]
Completely different hash
[New file content]

```

```

root@server01:~# sha256sum file1
777e06a9752b81e8991a3d057e9c70fa21234666f76915c54e4bd61a91fd0ce1 file1
[root@server01 ~]# echo "1" >> file1
[root@server01 ~]# sha256sum file1
463ce4a0d39b39cb3ca0b07a5b7531f7545ff9eb20e5e6b5bc45ef4fa0906830 file1
[root@server01 ~]#

```

*Calculating the hash value of a file.*

### SYNTAX

The syntax of the **sha#sum** commands is **sha#sum [options] [file name]**



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 12-15

## Discussing Backing Up, Restoring, and Verifying Data

### SCENARIO

Answer the following questions to check your understanding of the topic.

- 
1. **You and your team have devised a backup plan that includes a full backup once a month and then incremental backups between full backups. What are the two primary advantages of this backup scheme?**
  
  2. **Your team determines that for certain systems, snapshots are the appropriate backup method for critical storage contents. What is the advantage of a snapshot?**
  
  3. **What is the advantage of using an off-site backup and how can you transfer files from the main site to an off-site facility?**
  
  4. **There are several automated processes that copy files between servers for backups. These processes use the standard File Transfer Protocol (FTP). What is wrong about using FTP, and what is the better alternative?**
  
  5. **A senior-level administrator performed an audit on your backups and commended you for using SFTP to encrypt data as it traverses the network and the Internet. However, he introduces you to a new command, rsync, that has a particularly interesting advantage over SFTP. What is that advantage?**
-

# Activity 12-16

## Performing Integrity Checks on Files

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have synchronized individual prototype data files from your home directory to the data backup volume.

### SCENARIO

The R&D team is concerned about unauthorized users tampering with the prototype data. There's also the possibility that the data will become corrupted in a non-obvious way, which will compromise the integrity of the data. So, in order to be confident that the data hasn't changed, you'll run the files through a hash function and compare those hashes to hashes captured at a different time. If the hash values are the same, you can be assured of the data's integrity. If not, you'll know something went wrong.

1. Create hashes of the prototype files.
  - a) Enter **`sudo bash -c "sha256sum prototypes/* > hashes.txt"`**
  - b) Enter **`sudo cat hashes.txt`**
  - c) Verify that the text file lists five different hash values, each one corresponding to a specific .csv file.

```
61266b2d7504071517f155db57e0d5c8808c6dde88180993f29d52b6fcbae928 prototypes/gpu.csv
a49ed2f1b22bab6c02e7bd179a3b71a7be9ffcb2649dbc47960956e521554436 prototypes/hmd.csv
d38a12ace1a535f3e68635d6a4996590ecf20e51845a22239320167d8d1dae42 prototypes/ssd.csv
9c4b601e56b08b09a8979b1e373143c738d19ff091f55ff3fb2209cc9827a1f prototypes/stv.csv
f0874265115269d2c8b69d2a9ad2ed7f064efc496efe508e246d6b804bfa9289 prototypes/swatch.csv
```

2. Compare the captured hashes to the hashes of the current files.
  - a) Enter **`sudo sha256sum -c hashes.txt`**
  - b) Verify that the output indicates that all of the files are "OK".

In other words, **`sha256sum`** hashed the files, then compared those hashes to the list of hashes you generated in the previous step. The hashes are all identical, implying that the files haven't changed. In a production environment, you'd compare these hashes after some time had passed, after some potentially disruptive event, or right before resynchronizing.
3. Make changes to the files and verify that they fail the integrity check.
  - a) Enter **`sudo bash -c "echo GPU999 >> prototypes/gpu.csv"`**
  - b) Enter **`sudo rm prototypes/hmd.csv`**
  - c) Enter **`sudo sha256sum -c hashes.txt`**

- d) Verify that, this time, the integrity check failed on the file you modified, and that it could not find the file you deleted.

```
prototypes/gpu.csv: FAILED
sha256sum: prototypes/hmd.csv: No such file or directory
prototypes/hmd.csv: FAILED open or read
prototypes/ssd.csv: OK
prototypes/stv.csv: OK
prototypes/swatch.csv: OK
sha256sum: WARNING: 1 listed file could not be read
sha256sum: WARNING: 1 computed checksum did NOT match
```

---

## Summary

In this lesson, you implemented various security solutions, such as encryption, firewalls, IAM, and backups. These solutions will each contribute to the hardening of your Linux systems, reducing the risk of data compromise or other harm to the organization's assets.

**Do you plan to leverage a context-based permissions scheme like SELinux or AppArmor on your Linux systems? Why or why not?**

**Does your organization have off-site backups of critical data? If so, how does your organization conduct these backups? If not, do you think it's worth the effort?**



**Practice Question:** Additional practice questions are available on the course website.



# Lesson 13

## Working with Bash Scripts

**LESSON TIME: 3 HOURS**

### LESSON INTRODUCTION

Writing and using scripts are skills that are absolutely essential to being a Linux® administrator. Scripts greatly improve efficiency by minimizing the amount of repetitive typing you need to do at the CLI, and they also enable you to get more work done in a shorter amount of time. While the idea of scripting and programming in general may seem daunting to some administrators, Bash makes it surprisingly easy to work with scripts. In this lesson, you'll harness the power of scripts to make you much more productive on the job.

### LESSON OBJECTIVES

In this lesson, you will:

- Customize the Bash shell environment for script execution.
- Identify concepts fundamental to both scripting and programming.
- Write a simple Bash script and then execute it.
- Write more complex Bash scripts that incorporate flow control like conditional statements and loops.

# Topic A

## Customize the Bash Shell Environment



### EXAM OBJECTIVES COVERED

- 1.6 Given a scenario, configure localization options.
- 4.4 Given a scenario, analyze and troubleshoot application and hardware issues.
- 5.1 Given a scenario, deploy and execute basic Bash scripts.

You've been using the Bash shell, the default Linux shell, all throughout this course. Up until now, Bash has merely been a means of working at the CLI. However, there is much more you can do with Bash—for example, you'll start this lesson off by customizing the shell itself.

### SHELL ENVIRONMENT

The **shell environment** is the mechanism by which Bash, or any other shell, maintains settings and other behavioral details about the shell. The shell creates this environment when starting a session and uses the environment's settings to determine how to interact with the user.

The process of creating a new session is called **shell spawning**. This new session is a copy, and is called the child process. For example, the shell spawns a child process when the user enters a command. This child process becomes the new process and can also create more processes, which result in multiple generations of processes. Each process calls upon the shell environment and passes its details onto the next generation.

```

Initial shell process ID
[root@server01 ~]# ps -f
UID PID PPID C STIME TTY TIME CMD
root 12967 12960 0 14:01 pts/0 00:00:00 bash
root 13149 12967 0 14:02 pts/0 00:00:00 ps -f
[root@server01 ~]# bash
[root@server01 ~]# ps -f
UID PID PPID C STIME TTY TIME CMD
root 12967 12960 0 14:01 pts/0 00:00:00 bash
root 13190 12967 1 14:02 pts/0 00:00:00 bash
root 13232 13190 0 14:02 pts/0 00:00:00 ps -f
[root@server01 ~]#

```

*Using the Bash shell to spawn a child process.*

### SCRIPTS

A **script** is any computer program that automates the execution of tasks for a particular runtime or shell environment. Scripts are written in scripting languages, which are a subset of programming languages. Scripts typically do not have the feature set of a full-fledged program, but instead integrate with other programs and operating

system components to achieve automation. However, the terms are sometimes used interchangeably, and some languages can be used to write both scripts and full- featured programs.

## VARIABLES

**Variables** refer to entities whose values change from time to time. Most variables are set either by the operating system when you log in, or by the shell when it is initially invoked. Variables are the key components that comprise the shell environment. When you want to change the details of an environment, you change its variables and their values.

In Linux, variables can be categorized as shell variables or environment variables. **Shell variables**, by default, do not have their values passed onto any child processes that spawn after them. Environment variables, on the other hand, do get passed on.



**Note:** Shell variables are sometimes called local variables, and environment variables are sometimes called global variables. These names have different meanings in the context of scripting, however.

## VARIABLE SETTING AND REFERENCING

To set a shell variable, simply enter VAR=value such as MYVAR=123

In order to reference a variable, you must type it in the format \${VARIABLE NAME}

To retrieve the value of a variable, you can enter echo \${VARIABLE NAME} at the CLI. For example, echo \$SHELL will print your default shell (e.g., /bin/bash).

## ENVIRONMENT VARIABLES

An **environment variable** is a variable that is inherited from parent shell processes and is subsequently passed on to any child processes. An environment variable consists of a name, usually written in uppercase letters, and a value, such as a path name.

Within the environment, variables are referenced as key–value pairs in the format KEY=value and KEY=value1:value2 for variables with multiple values.

## DEFAULT ENVIRONMENT VARIABLES

Some of the default environment variables and their functions are provided in the following table.

| Environment           | Specifies                                            |
|-----------------------|------------------------------------------------------|
| HOSTNAME={hostname}   | The hostname of the system.                          |
| SHELL={shell path}    | The shell path for the system.                       |
| MAIL={mail path}      | The path where mail is stored.                       |
| HOME={home directory} | The home directory of the user.                      |
| PATH={user path}      | The search path.                                     |
| HISTSIZE={number}     | The number of entries stored in the command history. |
| USER={user name}      | The name of the user.                                |

## LOCALIZATION ENVIRONMENT VARIABLES

Environment variables can also be used to configure localization options, typically by editing the `/etc/locale.conf` file and assigning the appropriate locale to the variable. Some of these variables are described in the following table.

| Environment                  | Specifies                                                                                                                                                                                                                                                                                                                                                              |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>LC_*= {locale}</code>  | A collection of localization environment variables, including, but not limited to: <ul style="list-style-type: none"> <li><code>LC_ADDRESS</code> to set the postal address format.</li> <li><code>LC_MONETARY</code> to set the format of monetary values.</li> <li><code>LC_MEASUREMENT</code> to set the measurement system (e.g., metric vs. imperial).</li> </ul> |
| <code>LANG={locale}</code>   | The locale to use for all <code>LC_*</code> variables that aren't explicitly defined.                                                                                                                                                                                                                                                                                  |
| <code>LC_ALL={locale}</code> | The locale to use for all options, overriding any <code>LANG</code> and <code>LC_*</code> values. Typically used for troubleshooting purposes.                                                                                                                                                                                                                         |
| <code>TZ={time zone}</code>  | The system time zone. This is an alternative to using commands like <code>date</code> or <code>timedatectl</code> to set the time zone.                                                                                                                                                                                                                                |

## THE `export` COMMAND

You can effectively change a shell variable into an environment variable by using the `export` command. For example, if you have a shell variable `SHL_VAR`, you can enter `export SHL_VAR` to make it an environment variable.

You can also change the value of a variable while exporting it, including existing environment variables. You can do this by entering something similar to `export SHL_VAR="New value"` at the CLI. This will set the value for all child processes spawned from this shell.

```

root@server01 ~]# echo $SHL_VAR
[root@server01 ~]# export SHL_VAR="New value"
[root@server01 ~]# echo $SHL_VAR
New value
[root@server01 ~]#

```

No initial value

Create environment variable

Now has a value

Changing the value of a shell variable.

In order to set the value of an environment variable for all future Bash sessions, you can add an **export** statement to your **.bash\_profile** file. To automate this process for new users, and to ensure those with a similar job roles have the same environment variable settings, you can modify the **.bash\_profile** file in the **/etc/skel** directory. To set the value of an environment variable system-wide, add an **export** statement to the appropriate file in the **/etc/profile.d/** directory.

## SYNTAX

The syntax of the **export** command is **export [options] [NAME[=value]]**

## THE env COMMAND

The **env** command is used to run a command with modified environment variables. By supplying the name of a variable and a value in the key-value pair format, as well as supplying a command to run, you can change the value of the specified variable for that particular command session. If the variable does not exist, it will be added to the environment. Likewise, you can use the **-U** option to remove the specified variable from the environment in which the specified command runs. Consider using **env** if you want to override values in child processes or add new ones.

Issuing the command without any arguments will display all variables in the environment as well as their corresponding values.



*Note: The `printenv` command is functionally the same as issuing `env` without any arguments.*

```
root@server01 ~]# env
XDG_VTNR=1
XDG_SESSION_ID=1
HOSTNAME=server01
SHELL=/bin/bash
TERM=xterm-256color
HISTSIZE=1000
QTDIR=/usr/lib64/qt-3.3
OLDPWD=/root/Documents
QTINC=/usr/lib64/qt-3.3/include
QT_GRAPHICSSYSTEM_CHECKED=1
USER=root
```

Environment variables

*Listing environment variables and their values.*

## SYNTA

### X

The syntax of the **env** command is **env [options] [NAME=value] [command]**

## PRINTING ALL VARIABLES

You can use the **set** command without any arguments to print all shell variables, environment variables, and shell functions. This command can also enable the use of options in a shell script to change its behavior.

## COMPARING **export**, **env**, AND **set**

To summarize the difference between these three commands:

- **export**—Change the value of a variable for all child processes.
- **env**—View environment variables or change the value of a variable for a specified command.
- **set**—View shell variables or change the value of shell attributes.

## SEARCH PATHS

A **search path** is a sequence of various directory paths that is used by the shell to locate files. Paths can be assigned to the **PATH** environment variable. The **PATH** variable comprises a list of directory names separated by colons. You can add a new path to an existing group of path names, modify a path, or delete a path.

Usually, directories that contain executable files are assigned to the **PATH** variable. This enables you to enter the name of an executable at the CLI without needing to specify its full directory path. This is because the **PATH** variable searches its directories for the name of the executable.

## HISTFILESIZE

**HISTFILESIZE** is an example of a shell variable that enables you to set the maximum number of lines contained in the command history file. It also enables you to specify the number of lines to be displayed on running the **history** command. For example, by assigning a value of **20** to this variable, the history file gets truncated to contain just 20 lines. The default value of this variable is **1000**.

## THE **alias** COMMAND

The **alias** command is used to customize the shell environment by generating command-line aliases. Aliases are shorthand for longer expressions. Using aliases, you can create a short string that represents a longer command with various options and arguments. For example, you can create an alias called **myls** that executes the **ls - al** command.

The Bash shell maintains a list of aliases that you can view by using the **alias** command by itself. You can also remove aliases using the **unalias** command. By default, aliases are only maintained for the current shell and for the user that created them. To have them persist, add the appropriate **alias** command to **.bashrc** or **.bash\_aliases**, which is called by **.bashrc**.

The terminal window shows the following session:

```

root@server01 ~]# alias myls="ls -al"
root@server01 ~]# myls
total 112
dr-xr-x---. 19 root root 4096 Jan 9 15:37 .
dr-xr-xr-x. 21 root root 4096 Jan 2 18:41 ..
-rwxrwxr-x. 1 root root 1688 Dec 11 18:28 anaconda-ks.cfg
-rw-r--r--. 1 root root 133 Jan 9 15:23 archive.tar.bz2
-rw-r--r--. 1 root root 141 Jan 9 15:20 archive.tar.gz
-rw-r--r--. 1 root root 180 Jan 9 15:22 archive.tar.xz
-rw-----. 1 root root 6353 Jan 7 20:40 .bash_history
-rw-r--r--. 1 root root 18 Dec 29 2013 .bash_logout
-rw-r--r--. 1 root root 176 Dec 29 2013 .bash_profile
-rw-r--r--. 1 root root 176 Dec 29 2013 .bashrc
drwx-----. 17 root root 4096 Jan 2 14:56 .cache

```

Annotations point to "Alias name" and "Expression used by alias".

*Creating an alias for a command expression.*

## SYNTA

### X

The syntax of the `alias` command is `alias [alias name[='command with options']}`

## THE time COMMAND

The `time` command is used to gather information about how long it took to execute a command, as well as some additional statistics about the I/O and memory used in command execution. You provide the command you want to time as an argument to the `time` command. By default, the command outputs the following time statistics:

- The real time that elapses between the command's invocation and its termination.
- The user CPU time; i.e., the time spent running code in user space.
- The system CPU time; i.e., the time spent running code in kernel space.

The terminal window shows the following session:

```

root@server01 ~]# time ls -Rl /sys > /dev/null
real 0m0.883s
user 0m0.150s
sys 0m0.733s

```

Annotations point to "Real time elapsed", "Command to time", "User CPU time", "System CPU time", and "Suppress output".

*Timing a command.*

By finding out how long it takes a command to run, you can get a better idea of how to optimize frequent tasks. Some commands may complete the same task faster than other commands.



**Note:** Some shells, including Bash, provide their own `time` command as the default. To use the full command, reference its binary at `/usr/bin/time`.

## SYNTAX

The syntax of the `time` command is `time [options] {command}`

## ENVIRONMENT AND SHELL TROUBLESHOOTING

You may encounter some issues when using or customizing the Bash shell environment. The following are some troubleshooting tips.

- When adding an alias, check the syntax. For example: `ls='ls -la'`
- When executing scripts or other programs, if they are not stored in the normal locations for executable files, then add their location to the `PATH` variable or execute them with a `./` preceding the command.
- Use the `export` command to set a variable for all shell child processes.
- Configure environment variables in the `~/.bash_profile` file to make the variable available to all shells. For example, if a service account requires certain environment variables, you can set them in the `~/.bash_profile` for that account.
- Edit the `~/.bash_profile` file to change default variables.
- Ensure values are set for any environment variables that a software package has a dependency on. For example, if a Java application relies on the Java runtime environment, it may only be able to find and access that runtime environment if it is referenced in the `PATH` variable.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 13-1

## Discussing the Bash Shell

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. You are exploring your local (shell) variables on each system that you manage. The first variable you check is the SHELL variable. How can you check the SHELL variable and what do you expect it to display?
  
2. You have seen other administrators use the `export` command and you have seen it referred to in documentation for scripts. What does the `export` command do?
  
3. You log into a system that seems to have something incorrect in its configuration, but you are having trouble pinpointing why certain paths don't exist and why certain shell variables are set to non-standard values. How can you view all shell variables at once?
  
4. You believe either that the PATH variable is incorrectly set or that there are a significant number of commands missing from the system. How can you display the PATH variable value?
  
5. Another system administrator has aliased the `ls` command to `ls -la` and you don't like to see that much output in a simple `ls` command. How can you temporarily remove the alias?

# Activity 13-2

## Customizing the Bash Shell Environment

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

### SCENARIO

In order to enhance your productivity at the CLI, you decide to customize your Bash shell environment. For security reasons, you want to minimize the number of commands that are kept in the shell history, so you'll adjust the appropriate environment variable. You also plan on creating a directory to hold your future scripts, and in order to easily execute the scripts in that directory, you'll need to add it to your search paths. Lastly, as part of your auditing duties, you find yourself entering a rather lengthy command at the CLI every so often; this can get tedious, so you'll create a short alias for that command to make things easier.

1. Display the current environment variables.

- a) Enter `env`
- b) Verify that the current environment variables and their values appear on the screen.
- c) Verify that the `HISTSIZE` variable has a value of `1000`, indicating that a maximum of 1,000 of the most recently entered commands are stored in memory.

2. Reduce the maximum size of the command history by exporting its environment variable.

- a) Enter `echo $HISTSIZE` and verify that the variable has the expected value.
- b) Enter `export HISTSIZE=5`

**Note:** This value is intentionally low to make it easier to demonstrate.



- c) Enter more than five unique commands, one after another. For example, you could enter `echo 1`, `echo 2`, etc.
- d) Press the **Up Arrow** and verify that you can only return, at most, to the fifth-most recent command.

**Note:** You can revert the history size if you prefer, or you can log out and it will revert automatically.



3. Create a directory that will hold scripts.

- a) Enter `sudo mkdir /scripts`
- b) Enter the following:

```
sudo cp/opt/linuxplus/working_with_bash_scripts/
testscript.sh /scripts/testscript.sh
```

- c) Enter **testscript.sh**
- d) Verify that the command was not found.

**4.** Add **/scripts** as a search path to persist for the student account.

- a) Enter **echo \$PATH**
- b) Verify the current search paths that are set in this environment variable.
- c) Ensure you are in your home directory.
- d) Open **.bash\_profile** in the text editor of your choice.
- e) Scroll to the last line of the file and change it to say the following:

```
export PATH=$PATH:/scripts
```



**Caution:** Ensure you are appending `/scripts` to the `$PATH` variable, or you will overwrite the existing paths and be unable to easily enter many commands.

- f) Save and close the file.

**5.** Test that the path works as intended.

- a) Enter **source .bash\_profile** to reload your Bash profile and its variables.
- b) Open a terminal and enter **echo \$PATH** and verify that the new path was added to the end of the variable.

```
[student01@server01 ~]$ echo $PATH
/usr/lib64/qt-3.3/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/
bin:/sbin:/home/student01/.local/bin:/home/student01/bin:/home/student01/
.local/bin:/home/student01/bin:/scripts
```

- c) Enter **testscript.sh** and verify that the script executed successfully.

**6.** Create an alias for a lengthy command.

- a) Enter **lastlog | tail -n +2 | sort -k1**
- b) Verify that the list is sorted by user name, rather than the default of last login time.  
This is a somewhat cumbersome command to type over and over, so you'll create an alias to save time.
- c) Open **.bashrc** in the text editor of your choice.
- d) At the bottom of the file, on a new line, type the following:

```
alias ulog='lastlog | tail -n +2 | sort -k1'
```

- e) Save and close the file.
- f) Enter **source .bashrc**
- g) Enter **ulog** and verify that it produced the expected results.

# Topic B

## Identify Scripting and Programming Fundamentals



### EXAM OBJECTIVES COVERED

5.1 Given a scenario, deploy and execute basic Bash scripts.

Before you dive into scripting with Bash specifically, you need to familiarize yourself with some basic concepts that are shared by many scripting and programming languages used today. These concepts will lay the groundwork for writing your own scripts.

### BASH SCRIPTING

Not only is Bash the default shell in Linux, but it is also a powerful scripting language. Creating Bash scripts is incredibly useful in increasing the efficiency and productivity of your Linux administration tasks. Bash scripts can make Linux system calls and leverage existing tools in the user space. Essentially any program, tool, utility, or system function that you can call at the command-line you can also invoke in a Bash script. Likewise, Bash scripts support modern programming elements like loops and conditional statements to enhance the logic of the task(s) being automated.

### SYNTAX

Just as commands at the CLI have a syntax, so too do scripting languages. A language's **syntax** are the rules that define how you write the code. Each language has its own syntax, but many share a few commonalities. Because of its association with the underlying Linux operating system, the syntax of a Bash script is very similar to what you'd input line-by-line at a CLI.

### VARIABLE ASSIGNMENT

**Variable assignment** is the act of defining a variable as having a certain value. In code, you assign values to variable names. The values in a variable may change throughout the script's execution, but this is not required. The purpose of variables is to store values for later use, and to enable you to reference these values without explicitly writing them out in the code.

Many programming languages, like C, require you to define the type of variable before you assign it to a value. Examples of types include integers, floats, strings, and more. Essentially, these types define exactly what kind of information the variable holds.

However, you don't have to declare variable types in Bash. Instead, all Bash variables are treated as strings.

### BASH VARIABLE ASSIGNMENT

Bash variables are assigned as follows.

```
my_str='Hello, World!'
```

Note the lack of whitespace around the equals sign—this is a strict rule in Bash.

## VARIABLE SUBSTITUTION

The act of referencing or retrieving the value of a variable is called **substitution** or **parameter expansion**. After you assign a value to a variable, you reference that variable later in the code so that you don't need to hard-code values into the script's logic.

### BASH VARIABLE SUBSTITUTION

When referencing variables in Bash, you need to add a dollar sign (\$) at the beginning of the variable name:

```
echo $my_str
```

This will print "Hello, World!" to the console.

## COMMON OPERATORS

Operations enable you to perform some sort of task on the variables and values that you specify in your code. In most cases, this task is the evaluation of an expression.

**Operators** are the objects that can evaluate expressions in a variety of ways.

Operands are the values being operated on. There are many different kinds of operators that apply to most languages, including:

- **Arithmetic operators.** These include addition, subtraction, multiplication, division, and more advanced mathematical operations.
- **Comparison operators.** These include checking if operands are equal, if one operand is less than or greater than another operand, and more.
- **Logical operators.** These operators connect multiple values together so they can be evaluated, and include AND, OR, and NOT.
- **String operators.** These are used in operations that manipulate strings in various ways, including concatenating strings, returning a specific character in a string (slicing), verifying if a specific character exists in a string, and more.

Many languages find common ground when it comes to representing operators in code.

For example, in many languages, the == comparison operator evaluates whether or not the operands have equal values. Therefore, the expression 1 == 2 outputs to false. Note that this particular operator is distinct from a single equals (=), which is used in assigning values to variables.

However, some languages do not use the traditional symbols for comparison operators.

Instead, they use a letter-based syntax. For example, consider that the >= operator evaluates whether the left operand is greater than or equal to the right operand. In letter-based syntax, the operator is -ge. So, 1 -ge 2 outputs to false.

 **Note:** Despite all variables in Bash being strings, Bash permits arithmetic operations when variables are placed in the proper context.

## BASH OPERATIONS

The following is an example of an arithmetic operation in Bash. Note that expressions are evaluated when wrapped in double parentheses:

```
$((var1 + var2))
```

An example of a comparison operation in Bash. Note the use of square brackets and a letter-based operator:

```
[$var1 -ge $var2]
```

An example of a logical operation (AND) in Bash:

```
[$var1 -ge $var2] && [$var3 -le $var4]
```

An example of a string operation (concatenation) in Bash:

```
$var1$var2
```

## STRING LITERALS

A **string literal** is any fixed value that represents a string of text within source code. String literals are enclosed in single ('') or double ("") quotation marks. As long as you are using them consistently, using either single or double quotation marks is acceptable for basic string output. However, there are circumstances where double quotes won't preserve the literal value of all characters within the quotes.

For example, say you've defined the `my_str` variable mentioned previously. You then want to substitute this variable into a larger string literal, like so:

```
echo "My variable is $my_str"
echo 'My variable is $my_str'
```

The first line, because it is using double quotes, will print "My variable is Hello, World!" The second line, because it uses single quotes, will literally print "My variable is \$my\_str". Therefore, you must be careful to use the correct type of quotation mark depending on what your intent is.

## WHEN TO USE STRING LITERALS

It's not always necessary to use a string literal. If you don't wrap the previous `echo` example in quotation marks, then it will by default produce the same output as if you had wrapped it in double quotes. However, it's still good practice to wrap strings of text in quotes just to be sure. When you assign values with spaces in them to variables, you are required to use quotes.

## ESCAPE CHARACTER

In any language, Bash included, certain characters have special meaning. An **escape character** is used to remove that special meaning so the character can be used literally rather than interpreted as something else by the system. This is similar to using a string literal, but in the case of an escape character, you're only removing the special meaning from one character at a time.

In Bash, the escape character is a single backslash (\). For example, let's say you want to print a string to the command-line that actually contains a dollar sign. The dollar sign, as you know, has a special meaning—it is used in variable substitution. You can handle this by using single quotation marks, as so:

```
echo 'This $var is escaped'
```

Alternatively, if you wanted to use double quotes or no quotes at all, you could enter either of the following:

```
echo "This \$var is escaped"
echo This \$var is escaped
```

Notice how the backslash escape character precedes the dollar sign, which is the character you want to be interpreted literally.

## ARRAYS

An **array** is a collection of values. In other words, an array enables you to store multiple values in a single variable. This can make your code much easier to read and maintain. For example, you might want to perform a single mathematical operation on dozens of different values. Instead of creating a variable for each value, you can create

an array to simplify your code. Another benefit of arrays is that you can easily update their values throughout the code.

Arrays are ordered based on their indices. Most languages start an array with an index of 0. When you assign values to an array, you can usually perform a compound assignment to assign all values at once. The order you place each value in the compound assignment will determine its index—i.e., the first value will be at index 0, the second at index 1, and so on. Languages like Bash can also use individual assignment to assign specific values to each index one-by-one.

## BASH ARRAYS

Compound assignment in Bash arrays uses parentheses with each value separated by a space:

```
my_arr=(1 "Hello" 3.1)
```

Individual assignment adds a value to a specific index in brackets:

```
my_arr[0]=1
my_arr[1]="Hello"
my_arr[2]=3.1
```

You can reference an array by wrapping it in curly braces. You can reference a specific index of the array:

```
echo ${my_arr[0]}
```

This will print "1". You can also reference all of the values in an array by using the asterisk (\*) or at symbol (@) in place of the index:

```
echo ${my_arr[*]}
```

## FUNCTIONS

A **function** is a block of code that you can reuse to perform a specific task. This is useful in writing efficient code, as calling a function can save you from having to write out the same or similar code over and over. You can define your own functions that you can call in other parts of the script, or even call from other scripts.

Like variables, you define a function with a unique identifier. You use this identifier to reference the reusable code within the function.

## BASH FUNCTIONS

In Bash, there are two ways of writing functions. Both involve placing the desired code in between curly braces. The first method is:

```
function my_func {
code...
}
```

If you're familiar with object-oriented programming languages like C, you might be more comfortable with the second method:

```
my_func() {
code...
}
```

However, note that the open and closed parentheses are just there for visual clarity. In Bash, you don't pass in arguments to a function like you would with other programming languages. Instead, you pass in arguments similar to how you would at the command-line.

## COMMENTS

In the world of programming, comments are a method of annotating source code so that it is easier for the author and other programmers to understand. In most languages, comments are ignored by the system that compiles, interprets, or otherwise executes the program. They therefore exist as a way to document various elements of the code within the code itself.

In Bash, the number sign (#) indicates that every character after it on that line is part of a comment, and not to be executed. Although you are free to comment your code how you want, it's usually good practice to include one or more comment lines at the top of the script that explain what that script does, and to comment each line or code block that may require explanation. You should refrain from commenting on a line of code whose purpose is obvious, as too many comments can clutter the source code and can actually make it harder to understand.

### BASH COMMENTS

The following is an example of a short script with comments:

```
This script determines how many files are remaining to process
in a directory.
```

```
num_files=432 # current number of files processed
total_files=512 # total number of files to process

echo "There are $((total_files - num_files)) files remaining."
```

# Activity 13-3

## Identifying Scripting and Programming Fundamentals

### SCENARIO

Before you begin writing scripts to increase your productivity, you need to review some of the fundamentals of scripting and programming. So, you'll test your knowledge of these fundamentals to see if you're ready to begin scripting in earnest.

---

**1. Which of the following is the correct way to assign a variable in Bash scripting?**

- my\_name=Mary
- my\_name = 'Mary'
- my\_name Mary

my\_name= 'Mary'

**2. Which of the following is the correct way to substitute a variable in Bash scripting?**

- echo my\_name
- echo \$my\_name
- echo 'my\_name'
- echo (my\_name)

**3. True or false? Arrays start with the index 1**

- True
- False

**4. Which of the following is the correct way to substitute array index 2 in Bash scripting?**

- echo my\_arr[2]
- echo \$my\_arr[2]
- echo my\_arr{2}
- echo \${my\_arr[2]}

**5. Choose which of the following code statements accurately produces the following output:** The total is \$50.

- echo The total is \$50.
- echo "The total is \\$50."
- echo 'The total is \\$50.'
- echo "The total is \$50."

**6. What is the main purpose of a function in programming and scripting?**

- To ensure the program can terminate gracefully.
- To create a reusable chunk of code that performs a specific task.
- To evaluate arithmetic expressions.
- To spawn a new child process in the shell environment.

**7. What is the purpose of a logical operator in programming and scripting?**

- To connect values so that they can be evaluated together.
- To perform addition, subtraction, multiplication, and division.
- To concatenate strings.
- To check whether or not two operands are equal in value.

**8. Which of the following symbols indicates the start of a comment in Bash scripting?**

- ?
- #
- !
- \*

# Topic C

## Write and Execute a Simple Bash Script



### EXAM OBJECTIVES COVERED

5.1 Given a scenario, deploy and execute basic Bash scripts.

Now you're ready to begin writing and executing your own Bash scripts. In this topic, you'll implement some fundamental components of Bash scripting in order to create a simple working executable.

### **#!/bin/bash**

Bash scripts contain shell-specific instructions that may not be compatible with other Linux shells. This will result in a Bash script running on Bash shells correctly, while failing on other non-Bash shells in Linux. To specify that your script is written for the Bash shell, you need to add the line **#!/bin/bash** at the beginning of each script. This line will instruct the operating system to use the Bash shell interpreter when executing a script on an incompatible Linux shell.



**Note:** This line commonly called a "shebang."



**Note:** The interpreter name in this example is /bin/bash, which is the actual file name and location of the Bash interpreter.

### METACHARACTERS

**Metacharacters** are special characters that the Bash shell will, by default, interpret in a certain way. These are characters you must escape or enclose in quotes in order for them to be interpreted literally. The metacharacters in Linux are described in the following table. Several of these should look familiar.

| Metacharacter | Used In                                                                    |
|---------------|----------------------------------------------------------------------------|
| >             | Output redirection.                                                        |
| >>            | Output redirection (append).                                               |
| <             | Input redirection.                                                         |
| <<            | Input redirection (here documents).                                        |
|               | Piping.                                                                    |
| "             | Defining weak string literals.                                             |
| '             | Defining strong string literals.                                           |
| `             | Breaking out of a string literal to run the command between the backticks. |
| \             | Escaping characters.                                                       |
| =             | Variable assignment.                                                       |

| Metacharacter | Used In                                                                  |
|---------------|--------------------------------------------------------------------------|
| \$            | Variable substitution and other types of shell expansion.                |
| #             | Commenting.                                                              |
|               | Logical OR operations.                                                   |
| &&            | Logical AND operations.                                                  |
| *             | Wildcard matching (any number of characters matched). Nicknamed "splat." |
| ?             | Wildcard matching (single character matched). Nicknamed "hook."          |
| [ ]           | Wildcard matching (any characters between brackets matched).             |
| { }           | Parameter substitution and arrays.                                       |
| ( )           | Grouping commands.                                                       |
| &             | Running a process in the background.                                     |
| ;             | Separating multiple commands on the same line.                           |
| !             | Referencing command history. Nicknamed "bang."                           |

## EXIT CODES

An **exit code**, or **exit status**, is a value that a child process passes back to its parent process when the child process terminates. In the Linux world, a status code of 0 indicates that the process executed successfully. The exit code 1 or any number higher indicates that the process encountered errors while executing.

Many Bash scripts call upon other scripts or enable the user to leverage system commands with the script. Exit codes are useful because they can help these external entities detect whether or not initial script execution was successful, and then potentially change their behavior based on this exit code.

By default, a Bash script will generate an exit code of the last command that was run. You can also specify exit code behavior yourself. The exit code of the last run command is represented by the \$? special variable. You can, for example, redirect the exit code to standard output (stdout) and/or standard error (stderr). For example:

```
#!/bin/bash
chmod 888 file
echo $? >&2
```

This will redirect the exit code 1 to stderr. Likewise, you can use input redirection to take an exit code from standard input (stdin) into a Bash script.

## THE **exit** COMMAND

You can use the **exit** command in a script to force the shell to terminate with whatever exit code you provide. For example, **exit 1** will cause the script to terminate with a failure status. If you don't provide a number, **exit** will terminate with the exit code of the last command that was run.

## REDIRECTION AND PIPING

Just as you can take advantage of redirection and piping at the CLI, so too can you incorporate them in your scripts. Other than redirecting exit codes to stdout/stderr/stdin, you can also redirect data to and from files. For example, the following script

uses the `read` command to prompt a user for input, assigns that input to a variable, then appends the data to a file:

```
#!/bin/bash
echo 'What is your name?'
read name
echo $name >> name_list.txt
```

Likewise, you can pipe to other commands from within a script. The following example reads a text file of names (`cat`), pipes that text to search for a particular name (`grep`), then pipes that to a command that identifies the total count of that name (`wc`).

```
#!/bin/bash
cat name_list.txt | grep 'John' | wc -l
```

## SHELL EXPANSION

When a command is issued at the Bash shell, it is split into tokens, or words. **Shell expansion** is the process by which the shell identifies special tokens that it substitutes values for. Variable substitution is a type of shell expansion by which the shell identifies the `$` special character and then expands a variable into its actual value. In other words, in `echo $var`, the `echo` command doesn't "see" a variable; it sees whatever the value of that variable is when Bash expands it.

## ORDER OF EXPANSIONS

There are actually several more types of expansions—eight in total. Bash performs these expansions in a defined order, similar to an order of operations in a mathematical expression. That order is:

1. Brace expansion
2. Tilde expansion
3. Same time:
  - Parameter expansion/variable substitution
  - Arithmetic expansion
  - Command substitution
  - Process substitution
4. Word splitting
5. File/path name expansion

For the four expansions that happen at the same time, the expansion is done in left-to-right order as each appears.

## VARIABLE SUBSTITUTION WITH BRACES

As you've seen, the format `$var` is an expansion that will substitute a variable with its value. However, let's say you have the following code:

```
word=computer
echo "The plural of $word is $words."
```

This will print: **The plural of computer is .** This is because Bash expects a variable that is exactly named `$words` even though you just intended to add a letter to the actual value. You can get around this by enclosing the variable in braces, as such:

```
word=computer
echo "The plural of $word is ${word}s."
```

So, this will print: **The plural of computer is computers.**

## COMMAND SUBSTITUTION

**Command substitution** is a method of shell expansion in which the output of a command replaces the command itself. This is useful when you want to include a command's output within an existing string of text. For example:

```
echo "The current directory is `pwd`."
```

Notice that the command `pwd` is enclosed within backticks (`). Depending on what the CWD actually is, the output might be something like: **The current directory is /root.**

You can also use the format `$(command)` to perform command substitution, as in the following:

```
echo "The current directory is $(pwd)."
```



**Note:** The second format is preferred in Bash, as using backticks requires escaping certain characters (including nested backticks).

## GLOBBING

**Globbing** is another name for file/path name expansion. This method of shell expansion is used to replace a specific wildcard pattern with values that match the pattern. There are three special characters used in globbing: the asterisk (\*) used to match any number of characters; the question mark (?) used to match a single character; and characters within square brackets ([ ]) to match any of the characters listed.

The following are three examples of globbing:

```
cp *.txt ~/dest
cp ?.txt ~/dest
cp [abc].txt ~/dest
```

The first example copies any and all files with a .txt extension. This is because the wildcard character appears before the period, indicating that Bash should expand any possible combination of characters. The second example will only copy .txt files with a single character as a name, like **a.txt** and **b.txt**, but not **ab.txt**. The third example will only copy files named **a.txt**, **b.txt**, or **c.txt**.

## POSITIONAL PARAMETERS

A **positional parameter** is a variable within a shell script that is assigned to an argument when the script is invoked. For example, you can invoke a script `myscript.sh` like so:

```
myscript.sh arg1 arg2 arg3
```

The `arg1` argument corresponds to the positional parameter `$1`, `arg2` corresponds to `$2`, `arg3` corresponds to `$3`, and so on. Note that the space between arguments is used to separate positional parameters.

You can reference positional parameters directly in your scripts like you would any other variable:

```
#!/bin/bash
echo "The first argument is $1"
```

```
echo "The second argument is $2"
echo "The third argument is $3"
```

This is useful because your script can perform various operations on any arguments that are passed to it, as most scripts and commands do.

## SETTING POSITIONAL PARAMETERS

You can also set positional parameters directly in your scripts by using the **set** command. For example:

```
#!/bin/bash
set -- arg1 arg2 arg3
echo "The first argument is $1"
echo "The second argument is $2"
echo "The third argument is $3"
```

When this script is invoked without any arguments provided by the user, it will still have positional parameters **\$1**, **\$2**, and **\$3**, because they were set manually.

## THE exec COMMAND

The **exec** command is used to execute another command, replacing the current shell process with this new program's process (no new process is created). This can be useful when you want to prevent the user from returning to the parent process if an error is encountered. For example, you may want to terminate a privileged shell if a command fails.

You can also use the **exec** command without a command as an argument to redirect all output in the shell to a file. This is commonly used in scripts to suppress stdout at the CLI and instead send it only to one or more files. For example:

```
#!/bin/bash
exec > out.txt
pwd
ls -al
```

The current working directory and directory listing will output to **out.txt** and not the CLI.

## THE source COMMAND

The **source** command is used to execute another command within the current shell process. In this sense, it performs the opposite functionality of the **exec** command. This is useful when you'd like to stay within your current shell when executing a script. One example is sourcing a script that performs a change of directory (**cd**). After the script executes, your location will be whatever directory was changed to, whereas executing the script normally would keep you where you are.

Another situation where you might want to source a script is when your script changes or defines environment variables. For example, the following script (**export.sh**) exports a custom environment variable named **MYVAR**:

```
#!/bin/bash
export MYVAR=1
```

If you execute this script normally and issue the **env** command, you'll see that **MYVAR** is not listed. This is because the script spawned a new shell process, and once it terminated, its changes to the shell environment were destroyed. However, if you

enter the command `source export.sh` then the environment variable will be maintained because the script executes in your current shell.

## SCRIPT FILE EXTENSIONS

For the most part, file extensions in Linux are optional. Linux checks a file's metadata to determine what type of file it is. This goes for Bash scripts as well—you don't need to name your scripts with an extension. However, many developers have adopted the convention of adding `.sh` as an extension to their shell scripts, such as `myscript.sh`. While this does not imbue the script with any special meaning, it can make it easier for a person to identify that a file is indeed a script at a glance. Including a file extension can also help search operations in which you only want to look for or within shell scripts.

## SCRIPT PERMISSIONS

Remember, your ability to use any file, including a script, is constrained by the permissions assigned to that script. Even though you created the script, you won't automatically have permission to run it. You need to make sure two permissions are set for each user that needs to run the script:

- The execute (X) bit on the script itself.
- The write (W) and execute (X) bits on the directory containing the script.

You can set these permissions using `chmod` just as you would with any other file.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 13-4

## Discussing Writing and Executing a Simple Bash Script

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. You have written a log file for a script that you run to gather output, but each day the output gets overwritten. Which operator can you use to append rather than to overwrite the log file?
  
2. In your script, you want to echo the following message back to the user: "Hello, user. Please select an option to begin." You want "user" to be replaced by the name of the currently logged-in user. How can you do this no matter who is logged in?
  
3. After a storage device controller failure, you restore the repaired system from backups. One of the directories you restore has hundreds of files with extensions such as .ksh, .bsh, and .csh. Unlike Windows, Linux file extensions have nothing to do with what type of file it is. Why might the owner of the files have added these extensions?
  
4. You are editing an old backup script and find the line source directories.sh in the script. Displaying the contents of the script reveals a list of exported directory names as variables. What is the purpose of using the source command in the backup script?
  
5. What is the purpose of file globbing in scripts?

# Activity 13-5

## Writing and Executing a Simple Bash Script

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. Previously, you created a `/scripts` directory and added it to your PATH variable.

### SCENARIO

As part of managing the many storage partitions and volumes on your Linux servers, you routinely run a command like `df` to see if any devices are getting close to full. By monitoring the storage space being used by each device, you can avoid problems before they happen. However, entering this command over and over again is somewhat tedious, and it doesn't immediately retrieve the most relevant information in the most useful format. You want to be able to generate a more readable "dashboard" report of what storage devices are getting close to full, and which are fine. So, you decide to automate the process by writing a script to do the work for you.

1. Create the script file and give yourself the necessary permissions to execute it.
  - a) Enter `sudo touch /scripts/check_storage.sh`
  - b) Enter `sudo chown student## /scripts/check_storage.sh`
  - c) Enter `chmod 755 /scripts/check_storage.sh`  
You're giving yourself full access and everyone else read and execute permissions.
2. Set up your script editing environment.
  - a) From the desktop menu, select **Applications**→**Accessories**→**Text Editor**.  
You can write source code at the CLI, but it's often easier to write it in a visual editor, especially if you're new to programming/script writing.
  - b) Select **Open**→**Other Documents**.
  - c) Navigate to `/scripts` and open `check_storage.sh`.
  - d) On the bottom-right of the window, select the **Ln 1, Col 1** drop-down list.
  - e) Check the **Display line numbers** check box.
3. Begin the script by writing some contextual `echo` statements.
  - a) On line 1, type `#!/bin/bash`
  - b) Press **Enter** twice to skip to line 3.
  - c) Type `echo "Beginning storage check..."`
  - d) On new lines 5 and 6, type the following:



**Note:** Remember to replace `##` with your student number.

```
echo "Date: $(date)"
echo " ----- "
```

The first line will simply echo the current date and time. It does this by leveraging the **date** command using command substitution. The second line just makes the formatting a little more visually pleasing; you don't need type an exact number of hyphens.

**4.** Assign the main variables the script will use.

- a) On a new line 8, type **part=/dev/sda1**

You're defining this variable so you can use it later as the name of the partition to check.

- b) On a new line 9, type the following:

```
checkper=$(df -h | grep $part | awk '{print $5}' | cut -d
'%' -f1)
```

There's quite a bit being assigned to this variable. The following is a breakdown:

- First, the entire value is a command, so it uses the command substitution format, i.e., **\$(. ....)**
- The first subcommand uses **df** to get drive information.
- This is piped to the **grep** command, which searches the results for anything matching the **\$part** variable you just defined (in this case, **/dev/sda1**).
- The **awk** command extracts the data in the fifth column of these results. If you issue **df -h** by itself, you can see that the fifth column details the percentage of the storage device that is being used.
- Lastly, the **CUT** command simply strips the percent sign (%) from the value so that the script can perform arithmetic on it.
- The ultimate result is just a single number that represents the percentage of storage being used by the **/dev/sda1** partition.

**5.** Write **echo** statements that report storage usage and indicate the check is complete.

- a) On a new line 11, type the following:

```
echo "$part is $checkper% full."
```

- b) On a new line 13, type the following:

```
echo "Storage check complete."
```

- c) Select **Save**.

**6.** Test the script.

- a) Switch to a terminal, but keep the text editor open.
- b) At the terminal, enter **check\_storage.sh**

- c) Verify that the output displays the date and time, the percentage full message, and the completion message.

```
[student01@server01 ~]$./check_storage.sh
Beginning storage check...
Date: Wed Jan 2 16:00:41 GMT 2019

/dev/sda1 is 5% full.
Storage check complete.
```

**7. Redirect the pertinent output to a file instead of the CLI.**

- a) Return to the text editor.
- b) Place your cursor at the end of line 3 and press **Enter** twice.
- c) On a new line 5, type the following:

```
exec >> ~/storage_report.txt
```

Now, all output in this script will be redirected to a file, unless otherwise specified.

- d) Change the **echo** statement on line 15 so that it reads:

```
echo "Storage check complete. Report saved to
storage_report.txt." >&2
```

This will redirect the message to the CLI (through stderr) in order to bypass the **exec** command.

- e) Save the script.

**8. Test the script again.**

- a) From a terminal, run the script again.
- b) Verify that the only messages printed to the CLI are the beginning and completion messages.
- c) Enter **cat storage\_report.txt** and verify that everything else was sent to this file.

```
[student01@server01 ~]$ cat storage_report.txt
Date: Wed Jan 2 16:03:09 GMT 2019

/dev/sda1 is 5% full.
```

**9. From a functionality perspective, how does this script fall short? How could it be improved?**

# Topic D

## Incorporate Control Statements in Bash Scripts



### EXAM OBJECTIVES COVERED

5.1 Given a scenario, deploy and execute basic Bash scripts.

Some scripts can remain simple, but the true power of scripting comes from being able to control the flow of logic as it executes. In this topic, you'll augment your scripting skills through the use of conditional statements and loops.

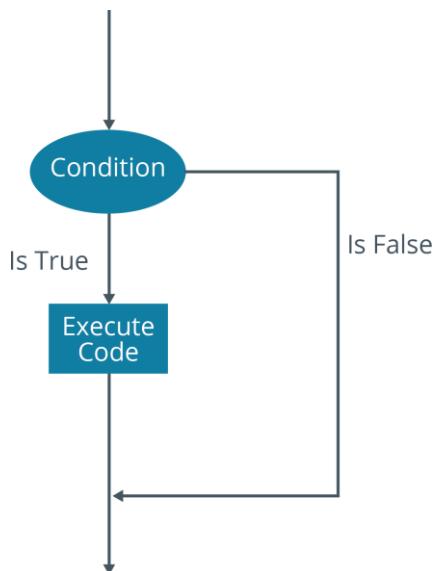
### LOGIC AND CONTROL STATEMENTS

A script's logic determines how it will process written code during execution. In Bash, as in most languages, there are various ways to design the logic of the code to essentially accomplish the same results in execution. Logic is therefore important in maximizing the efficiency and readability of code.

One of the most important components for implementing programming logic is a **control statement**. A control statement begins a section of code that will define the order in which instructions are executed. By controlling the flow of these instructions, you can write scripts to follow one or more paths based on certain circumstances.

### CONDITIONAL STATEMENTS

A **conditional statement** is a control statement that tells the program it must make a decision based on various factors. If the program evaluates these factors as true, it continues to execute the code in the conditional statement. If false, the program does not execute this code.



*The basic flow of a conditional statement.*

Conditional statements are fundamental to most programs and scripts, as they help you control the flow of executed code. For example, if a user enters some input, you might want to process that input differently based on a number of factors. The user might supply one argument and not another. Rather than executing the script as if all possible arguments were intended, you'd only execute the script with the argument the user supplied.

## THE if STATEMENT

In most languages, including Bash, the primary conditional statement is the **if** statement. An **if** statement contains a condition to be evaluated and one or more actions to be performed, if the condition is satisfied. If the condition is not satisfied, the actions are skipped and the next statement in the script is executed. In Bash, the end of the set of instructions is indicated by the **fi** statement.

The following is an example of a simple **if** statement:

```
var=5
```

```
if [$var -gt 1]
then
 echo "$var is greater than 1!"
fi
```

The **if** statement includes, between two square brackets, a condition to be evaluated. In this case, it's whether or not the **\$var** variable is greater than 1. On the next line is the **then** statement, within which is the code that will be executed if the prior condition is true. Lastly, the **fi** statement indicates the end of the entire **if** statement.

Because 5 is greater than 1, the message will echo to the screen. If it were not true, then nothing would happen.

## SYNTAX

The basic syntax of an **if** statement is as follows:

```
if [<condition to be evaluated>] then
 <code to execute if condition is true>
fi
```

## THE if...else STATEMENT

The **if...else** statement enables a choice between two actions based on the evaluation of a condition. If the condition is satisfied, the first action is performed; otherwise, the action following the **else** segment is performed. If there are more than two sets of instructions, one or more **elif** statements may be used to specify alternative sequences of action.

The following is an example of a simple **if...else** statement:

```
var=1
```

```
if [$var -gt 1]
then
 echo "$var is greater than 1!"
else
```

```
echo "$var is less than or equal to 1!"
fi
```

The value of \$var has changed since the previous example, which means that the first echo command won't execute. Rather than nothing happening, the else statement specifies what will happen if the condition is false: in this case, it is to print a different message to the screen.

## SYNTAX

The basic syntax of an **if...else** statement is as follows:

```
if [<condition to be evaluated>]
then
 <code to execute if condition is true>
else
 <code to execute if condition is false>
fi
```

The basic syntax of an **if...elif** statement is as follows:

```
if [<condition to be evaluated>]
then
 <code to execute if condition is true>
elif [<other condition to be evaluated>]
then
 <code to execute if other condition is true>
fi
```

## EXIT CODES WITH CONDITIONAL STATEMENTS

Conditional statements like **if...else** are good at handling process exit codes. For example:

```
chmod 888 file 2> /dev/null
```

```
if [$? -eq 0]
then
 echo "Permissions set successfully."
 exit 0
else
 echo "Could not set permissions."
 exit 1
fi
```

If the **chmod** command exits with a code of 0, then the success message is echoed to the screen, and any other process that executes this script will receive that exit code because of the **exit** command. Likewise, if the exit code is 1, then a custom error message is echoed to the screen. The default error message is suppressed, as it is being redirected to the null device.

## THE case STATEMENT

There may be times when you want to evaluate numerous conditions, such as a variable that can hold many different values, and each value requires its own action. You could define multiple **elif** branches in an overall **if** statement, but this can make your code difficult for a human to parse. The **case** statement helps you avoid this issue.

The following is an example of a simple `case` statement:

```
var=blue

case $var in
 red)
 echo "Your color is red."
 ;;
 green)
 echo "Your color is green."
 ;;
 blue)
 echo "Your color is blue."
 ;;
 *)
 echo "Your color is neither red, green, nor blue."
 ;;
esac
```

The first line in the `CASE` statement defines what variable it is that you're evaluating. Below that is the first condition, `red`, which has a closing parenthesis to indicate the end of the condition. On the next line is the action that will be performed if the color is indeed red—a message will display on the screen saying as much. The double semicolons (`;;`) indicate the end of the action.

This pattern is repeated, and can go on for as many conditions as you'd like. In this case, the last condition uses a wildcard (\*) to indicate that if the variable doesn't match any of the conditions above, then the following action will execute. The `esac` statement ends the `case` statement.

## SYNTAX

The basic syntax of a `case` statement is as follows:

```
case <variable> in
 <first condition>
 <code to execute if first condition is true>
 ;;
 <second condition>
 <code to execute if second condition is true>
 ;;
esac
```

## THE test COMMAND

The `test` command is used to check conditional logic and perform comparisons. You can use the `test` command in your shell scripts to validate the status of files and perform relevant tasks. It evaluates a conditional expression or logical operation and displays an exit status. The exit status is `0` if the expression is true and `1` if the expression is false.



**Note:** These codes are different than the exit codes generated upon process termination.

For example:

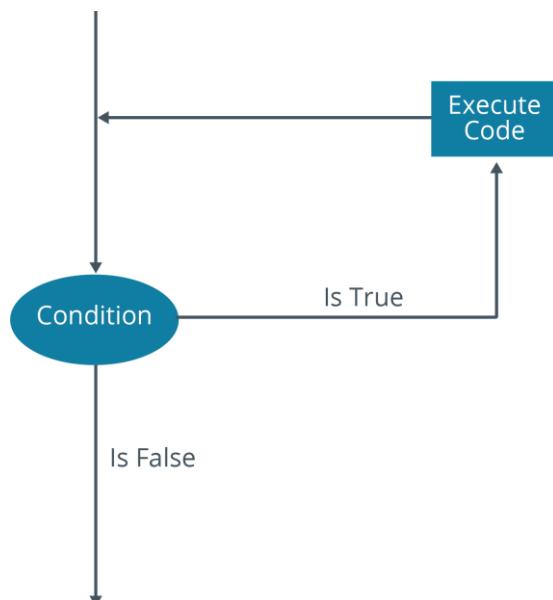
```
var=/etc

if test -d $var;
then
 echo "The $var directory exists!"
fi
```

This example uses the **-d** option to test if a directory exists. There are many such conditional options you can use. Consult the man page for the **test** command to see them all.

## LOOPS

Aside from conditional statements, another useful way to control the flow of logic in a script's code is by implementing loops. A **loop** is any control statement that executes code repeatedly based on a certain condition. In general, loops are a great way to keep a certain block of code active until no longer needed. There are three types of loops supported by Bash: the **while** loop, the **until** loop, and the **for** loop. All three types of loops are enclosed within the **do** and **done** statements.



*The basic flow of a loop.*

## THE while LOOP

The **while** loop enables you to repeat a set of instructions while a specific condition is met. The expression is evaluated, and if the expression is true, the actions in the loop are performed. The execution returns to the beginning of the loop and the expression is evaluated again. If the expression becomes false at any point, the execution breaks out of the loop and continues to the next block of code in the script.

The following is an example of a simple **while** loop:

```
var=1
```

```
while [$var -le 5]
do
```

```

 echo "The current number is $var."
 ((var++))
done

```

In this case, the condition being tested is whether or not the variable `$var` is less than or equal to `5`. As long as it is (i.e., the expression is true), then the code under `do` will execute. Below the `echo` command is an iterator, which simply adds 1 to the variable. This is common in any kind of loop, as without it, `$var` will always equal `1` and will therefore never break out of the loop.

## SYNTAX

The basic syntax of a `while` loop is as follows:

```

while [<condition to be evaluated>]
do
 <code to execute while condition is true>
done

```

## THE until LOOP

The `until` loop is similar to the `while` loop, except that the code is executed when the control expression is false. For example:

```

var=1

until [$var -ge 5]
do
 echo "The current number is $var."
 ((var++))
done

```

The condition in this loop is whether or not `$var` is greater than or equal to `5`. So, the code will execute until `$var` becomes `5`, at which point it will break out of the loop.

## SYNTAX

The basic syntax of an `until` loop is as follows:

```

while [<condition to be evaluated>]
do
 <code to execute while condition is false>
done

```

## THE for LOOP

The `for` loop executes a block of code as many times as specified by a numerical variable that is within the conditional part of the statement. Unlike a `while` or `until` loop, a `for` loop does not depend upon a condition being evaluated to false or true for it to stop or never begin in the first place. So, `for` loops are meant to always execute code a given number of times. This makes them ideal for processing iterable objects like arrays.

The following is an example of a simple `for` loop:

```
var=("Carla" "Robert" "Mary")
```

```
for i in ${var[*]}
```

```
do
 echo "$i is a member of the team."
done
```

The **for** statement evaluates every value in the `$var` array. In this case, each value is represented by the iterator `i`, though you can call this whatever you want. It's common to name it something similar to the variable you're looping through, or to simply call it `i` for iterator.

Then, the loop itself will execute three times—one for each value in the array. So, each person's name will be echoed to the screen.

## SYNTAX

The basic syntax of a **for** loop is as follows:

```
for i in <variable to loop through>
do
 <code to execute a specific number of times>
done
```

## LOOPING THROUGH RANGES

The **for** loop is also used to step through a range of numbers. These ranges are enclosed in curly braces, with the range itself indicated by two periods (`..`). For example:

```
for i in {1..5}
do
 echo "The current number is $i."
done
```

So, this loop will iterate exactly five times.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 13-6

## Discussing Control Statements in Bash Scripts

### SCENARIO

Answer the following questions to check your understanding of the topic.

---

- 1. What condition does the `while` loop test for?**
  
  - 2. You are creating a script that tests several conditions. Depending on the test outcome, you want one of three things to happen: send the results to a file, execute another script, or send all output to `/dev/null`. Which type of conditional statement can you use for this script?**
  
  - 3. You have created a script to check the existence of a file. If the file exists, you want to have the script echo the word "Success" and then exit with an exit code of 0. Otherwise, it should echo "Doesn't exist". Where in the script should you place the `exit 0` code?**
  
  - 4. True or false? There is a limit to the number of conditions in a case statement.**
  
  - 5. Your colleague has written a script with an array called `names` that holds the names of various users. In the script, a `for` loop iterates through each value in the array. The script is supposed to echo each name to the CLI, but instead, it only echoes the first name in the array multiple times. You confirm that the `for` loop has the proper syntax, so the problem must lie elsewhere. What do you think the problem is with the script?**
-

# Activity 13-7

## Incorporating Conditional Statements in Bash Scripts

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have `check_storage.sh` open in a text editor.

### SCENARIO

You want to make your script more useful to the administrators who will be receiving the reports. You can do this by enabling the script to make decisions based on various conditions. So, you'll use various **if** statements to output a different message for when the storage device meets certain thresholds of percentage full. Devices that are very close to full will trigger an urgent message, whereas those that are less full will trigger less urgent messages.

1. Return to `check_storage.sh` in the text editor.
2. Start writing the first conditional branch.
  - a) Place your cursor on the blank line 12.
  - b) Press **Enter**.
  - c) On a new line 13, type the following:  
`if [ $checkper -ge 95 ] && [ $checkper -le 100 ]`
  - d) Press **Enter**.
  - e) On a new line 14, type **then**
  - f) Place your cursor at the beginning of line 15 and press **Spacebar** four times.  
 You're not required to indent or create whitespace, but it helps make the code more readable.
  - g) Modify the **echo** statement on line 15 to read like the following:  
`echo "ALERT: $part is $checkper% full! Recommend immediate action!"`

You've just created the first **if** branch. This code checks to see if the percentage full value is greater than or equal to 95 and less than 100. If it is, then the script will echo an alert to the `storage_report.txt` file. However, you still need to write more branches to handle other conditions.

3. Write the next conditional branch.
  - a) Place the cursor at the end of line 15 and press **Enter**.
  - b) On a new line 16, type the following:  
`elif [ $checkper -ge 50 ] && [ $checkper -lt 95 ]`
  - c) Press **Enter**.

d) On a new line 17, type **then**

e) Press **Enter**.

f) On a new line 18, indent and then type the following:

```
echo "CAUTION: $part is $checkper% full! Consider freeing
up some space."
```

If the previous condition is not met, the script will move on to evaluating the condition in this **elif** branch. The condition here checks to see if the percentage full is greater than or equal to 50 *and* less than 95. If it is, then a different message will be echoed to the report file.

**4.** Finish writing the remaining conditional branches.

a) Press **Enter**.

b) Starting on a new line 19, type the following code:

```
elif [$checkper -lt 50]
then
 echo "$part is $checkper% full. No action needed."
else
 echo "Encountered an error. Status code: $" >&2
 exit $?
fi
```

The next branch will output another message if the percentage full is less than 50. If none of these conditions are met (i.e., the percentage value is above 100 or it isn't a number), then the last **else** branch will throw an error. That exit code will be printed to the CLI and the script will terminate with this code.

c) Save the script.

**5.** Test the script to see if the conditions work as expected.

a) From a terminal, enter **check\_storage.sh**

b) Enter **cat storage\_report.txt** and verify that, because **/dev/sda1** is not very full, the report indicates that no action is needed.

```
[student01@server01 ~]$ cat storage_report.txt
Date: Wed Jan 2 16:03:09 GMT 2019

/dev/sda1 is 5% full.
Date: Wed Jan 2 16:08:12 GMT 2019

/dev/sda1 is 5% full. No action needed.
```

In other words, the script chose the correct action to take based on the conditions you set.

**Note:** You'll test some of the other conditions in the next activity.



# Activity 13-8

## Incorporating Loops in Bash Scripts

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. You have `check_storage.sh` open in a text editor. You also have a 2 GB volume that is mounted on `/backup/sys`

### SCENARIO

Your script is coming along, but it still needs improvement. You want to be able to output the status of all relevant storage partitions/volumes on the system, not just one or a few. You need a way to programmatically test your conditions for each device, rather than hardcode device names in your script—especially if the storage devices are likely to change. So, you'll leverage a `for` loop to iterate over each recognized storage device to perform the necessary checks.

1. Adjust the `part` variable so that it holds an array of device names.

- a) Place your cursor on line 10 where the `part` variable is defined.
- b) Change this line to the following:

```
part=$(df -h | awk '{print $1}' | grep '/dev')
```

This is similar to the `checkper` variable. The difference is, it will extract all text that is in the first column (device name), and then filter by devices that start with `/ dev` to exclude temporary file systems. The `part` variable therefore becomes an array that holds all permanent storage device names on the system.

2. Insert a `for` loop that will iterate through the `part` array.

- a) Place your cursor at the end of line 10, then press **Enter** twice.
- b) On a new line 12, type the following:

```
for i in ${part[*]}
```

This begins the `for` loop. The `i` variable is the iterator. The `part` variable is being referenced as an array, with the asterisk (\*) indicating all values in that array. For every index in the array (i.e., every device name), the script will execute what follows.

- c) Press **Enter**, and on line 13, type `do`  
This begins the code that the loop will execute on each iteration.
- d) Place your cursor at the end of line 28 and press **Enter**.
- e) On line 29, type `done`

This terminates the `for` loop. The conditional statements within this loop will be executed for each iteration.

3. Change `$part` references to use the iterator instead.

- a) On line 14, change the `grep $part` portion of the command to `grep $i`

You need to get the information for each device individually. This means you need to reference the iterator, not the entire array.

- b) On line 18, change the **\$part** reference to **\$i**
  - c) Do the same for lines 21 and 24.
- 4.** Clean up the source code.
- a) Highlight all of lines 14 through 28.
  - b) Press **Tab** to indent the selected lines. Again, this makes the code more readable.
  - c) Save the file.
- 5.** Test the script.
- a) From a terminal, enter **check\_storage.sh**
  - b) Enter **cat storage\_report.txt**
  - c) Verify that the report lists all storage devices and their appropriate warning messages.

```
CAUTION: /dev/mapper/centos-root is 51% full! Consider free
ing up some space.
/dev/mapper/backup-logbk is 1% full. No action needed.
/dev/sda2 is 23% full. No action needed.
/dev/sdal is 5% full. No action needed.
/dev/mapper/centos-home is 6% full. No action needed.
/dev/mapper/backup-sysbk is 4% full. No action needed.
ALERT: /dev/sr0 is 100% full! Recommend immediate action!
```

- 6.** Simulate a volume becoming full, then test the script again.
- a) At a terminal, enter the following:
- ```
sudo dd if=/dev/zero of=/backup/sys/test bs=1M count=1100
```
- b) Verify that roughly 1.2 GB was copied to the volume.
 - c) Run your script again and read the report.
 - d) Verify that, this time, you receive a caution warning because the volume is past 50% full.
 - e) Enter the following:
- ```
sudo dd if=/dev/zero of=/backup/sys/test2 bs=1M count=800
```



*Caution: The output file name and count have both changed.*

- f) Run the script again and view the report.
- g) Verify that you received the most urgent message for the volume.

- 7.** Close the text editor, but keep the terminal open.
-

## Summary

In this lesson, you customized the Bash shell and performed various operations in it. You also created and executed Bash scripts. This will enable you to more efficiently perform your job as a Linux administrator.

**What environment variables might you configure in your shell environment?**

**What are the various tasks that you might perform in your environment by running a shell script?**



**Practice Question:** Additional practice questions are available on the course website.



# Lesson 14

## Automating Tasks

**LESSON TIME: 1 HOUR**

### LESSON INTRODUCTION

Bash scripting goes a long way toward automating your administrative duties on Linux®. But there are plenty more methods of automating the tasks you perform every day. These methods can further improve your productivity and lead to an overall boost in task accuracy and usefulness. As time goes on, you'll rely less and less on manual administration.

### LESSON OBJECTIVES

In this lesson, you will:

- Run jobs on a set schedule.
- Implement version control for files using Git.
- Identify concepts fundamental to orchestration.

# Topic A

## Schedule Jobs



### EXAM OBJECTIVES COVERED

2.6 Given a scenario, automate and schedule jobs.

Some tasks need to be performed repetitively on a set schedule. By scheduling these tasks, you won't need to remember to execute them manually, and you won't need to be able to access the system at the given times.

### THE **at** COMMAND

The **at** command is used to run a task once, at a specified time. It is not designed for repetitive or regularly scheduled tasks. The **at** command is very flexible. Users can specify a particular date and time, or cause the scheduled command to run after a given period of time.

The command is typically used in an interactive manner, where the **at** command and time interval are specified, then a task is defined in an interactive prompt. This enables the user to enter a path to a script or a command to be run. Pressing **Ctrl+D** exits the interactive mode.

```
[root@server01 ~]# at now + 1 minute
at> sendmail root@server01 "Hello"
at> <EOT>
job 6 at Wed Jan 9 16:40:00 2019
[root@server01 ~]# date
Wed Jan 9 16:40:02 GMT 2019
You have new mail in /var/spool/mail/root
[root@server01 ~]#
[root@server01 ~]#
```

*Running a command at a specified time.*

### SYNTAX

The syntax of the **at** command is **at [options] {time}**

### at COMMAND OPTIONS

Some of the **at** command options are described in the following table.

| Option         | Used To                                                             |
|----------------|---------------------------------------------------------------------|
| -m             | Send mail to the user when the job completes, regardless of output. |
| -M             | Prevent mail from being sent to the user.                           |
| -f {file name} | Read a job from a file rather than standard input.                  |
| -t {time}      | Run the job at the specified time value.                            |
| -v             | Display the time the job will be executed.                          |

## TIME SPECIFICATIONS

The **at** command takes several possible arguments for specifying time. Examples include:

- **noon** to specify 12 P.M.
- **teatime** to specify 4 P.M.
- **midnight** to specify 12 A.M.
- **now + 3 minutes** to specify the time three minutes from now.
- **now + 1 hour** to specify the time one hour from now.

## RELATED COMMANDS

The **atq** command can be used to view the current queue of tasks scheduled by the **at** command. The **atrm** command can be used to delete a scheduled task.

## CRON JOBS

Administrators and users may want to have scripts or commands execute on a regular basis. Automation of these kinds of tasks is efficient and consistent. The **CRON** daemon is used to manage these scheduled tasks called **cron jobs**.

The **CRON** daemon checks its **crontab** configuration file each minute to discover whether there are any tasks to be accomplished. If there are, it executes them. If there are not, it goes back to sleep until the next minute.

Cron jobs can be used to specify tasks each minute, hour, day, month, and any day of the week. This makes them extremely flexible.

## THE crontab COMMAND

The **CRON** daemon is controlled using the **crontab** command. The command assumes the current user unless the **-U** option is specified. You can create, view, and delete **crontab** files using the **crontab** command.

Some of the options of the **crontab** command include:

| Option | Used To                                                       |
|--------|---------------------------------------------------------------|
| -e     | Edit the <b>crontab</b> file for the current user.            |
| -l     | View the <b>crontab</b> file for the current user.            |
| -r     | Delete the current <b>crontab</b> file.                       |
| -u     | Create a <b>crontab</b> file on behalf of the specified user. |

```
[root@server01 ~]# crontab -l -u student01
00 08 * * * /bin/echo "Please fill in your time sheet."
[root@server01 ~]#
```

*Viewing a user's crontab file.*

## SYNTAX

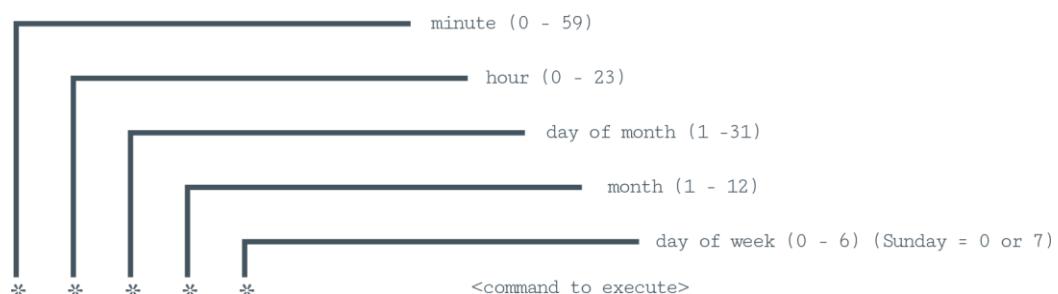
The syntax of the **crontab** command is **crontab [options] [file/ user]**

## THE crontab FILES

The **crontab** files are referenced by the **cron** daemon to determine what scheduled tasks might exist. The **crontab** files are managed using the **crontab - e** command. This command opens a text editor (Vim by default on most systems), enabling users to specify when they want a task run, and what task it is.



**Note:** Do not use a regular editor for managing the *cron* daemon. The **crontab -e** command includes other features for managing the *cron* daemon. For example, after editing the file, it automatically restarts *cron*.



*The format of crontab entries.*

The following are examples of lines in a **crontab** file that schedule tasks at certain times:

\* 20 \* \* 1-5 /path/to/command —executes the command at 8 P.M., Monday through Friday.

15 2 \* \* \* /path/to/command —executes the command at 2:15 A.M., daily.

30 4 1 \* \* /path/to/command —executes the command at 4:30 A.M. on the first day of each month.

## CRON DIRECTORIES

The **crontab** files that contain scheduled task information can be found in the **/etc/cron.d/** directory and in the **/var/spool/cron/** directory. The root user can schedule system-wide tasks by using the **/etc/cron.d/** directories. Services may also add scheduled tasks at this location. The root user can add tasks to the directory or can add tasks to the **/etc/crontab** file.

Regular users are not allowed to populate the **/etc/cron** directories, so each standard user can schedule their own tasks in a personal directory located at **/var/spool/cron**. Any tasks listed here will execute with the standard user's credentials.

The **/etc/** directory also contains several default cron directories that administrators can use to place scripts to be executed on a regular basis. These directories are **/etc/cron.hourly**, **/etc/cron.daily**, **/etc/cron.weekly**, and **/etc/cron.monthly**. Link or copy a script into these directories to use their schedule to run your commands.

Some Linux distributions pre-populate the **/etc/crontab** file with particular tasks. You may find that **logrotate**, **tmpwatch**, **rkhunter** (Rootkit Hunter), etc., may already be present. The installation of these kinds of services may also include the creation of **/etc/crontab** entries.

 **Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 14-1

## Discussing Scheduling Jobs

### SCENARIO

Answer the following questions to check your understanding of the topic.

---

1. **You are scheduling new backup tasks on every new Linux system that your team configures and manages. Each requires its key directories to be backed up. Would the `at` command or a cron job be best for scheduling the backup script? Why?**
  
  2. **Anyone can schedule cron jobs. As a regular user, you need to schedule a file transfer from your local workstation to the shared directory on one of the development servers. You need to create a new cron job. Which command can you use to create the cron job?**
  
  3. **As the root user, you need to check all scheduled tasks that belong to root. How can you check them?**
  
  4. **How can you remove an entry from the `at` command queue?**
  
  5. **You have added several jobs to the `at` queue and you need to check the status. How can you check the `at` queue?**
-

# Activity 14-2

## Scheduling a Single Task

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

### SCENARIO

Periodically, the developers at Develetech need to execute a task after hours. The schedule is not predictable and they need to be able to manage these tasks themselves. You will use the **at** command to satisfy this requirement.

---

Schedule a task to run for two minutes into the future from your current time.

- a) In your home directory, use the **touch** command to create a file named **fileA**
  - b) Check the current time on your system.
  - c) Enter **at now + 2 minutes** to access the interactive mode of the command.
  - d) Enter **rm -f ~/fileA** and then press **Ctrl+D** to return to Bash.
  - e) Enter **atq** to view the scheduled job.
  - f) After two minutes, ensure that the command executed by checking the contents of your home directory to see if **fileA** was removed.
-

# Activity 14-3

## Scheduling Repeated Tasks

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

### SCENARIO

Develetech adopted a new policy that requires all users to fill in their time sheets every day. You'll create a daily reminder for all user systems.

1. Schedule a cron job to email a reminder every day at a specified time.
  - a) Enter **sudo crontab -u cmason -e** to specify a cron job for Chris Mason.
  - b) Verify that Vim opens a temporary file automatically.
  - c) Type the following line in the file:  
**<MM> <HH> \* \* \* /bin/echo "Please **fill** in your time sheet."**  
 Replace <MM> and <HH> with the appropriate minute and hour time values in 24-hour time format. Ensure that the time you enter is three minutes ahead of the current system time. This way, you'll be able to see the message in class.  
 For example, if the time is 2:30 P.M., you'd type:  
**33 14 \* \* \* /bin/echo "Please **fill** in your time sheet."**
  - d) Save and close the file.
  - e) From the desktop menu, select the icons at the top-right, then select **student##→Log Out**.
  - f) Select **Log Out**.
2. Verify that Chris Mason received the reminder for the scheduled job.
  - a) Log in as **Chris Mason**.
 



**Note:** You can ignore the **Welcome** screen, or you can step through the wizard to dismiss it.

    - b) Open a terminal.
    - c) Wait for the time to pass for the cron job to execute.  
 Remember, you can use **date** to check the time. You can also check the time from the desktop menu in the GUI.
    - d) Enter **mail**
    - e) Enter **1** to read the contents of the first email message.
    - f) Verify that the mail contains a reminder to fill in the time sheet.
    - g) Press **q** to quit the mail service.
    - h) Log out as **Chris Mason** and log back in as your student account.

# Topic B

## Implement Version Control Using Git



### EXAM OBJECTIVES COVERED

5.2 Given a scenario, carry out version control using Git.

If you or your colleagues are developing software, they'll need some place to store and manage the source code. This can quickly become an overwhelming task, especially when many versions of the code are involved. In this topic, you'll automate code management and version control through a program called Git.

### GIT

**Git** is a distributed version control system primarily used by developers who are collaborating on projects. Git was developed by Linus Torvalds, the same person who created Linux. Git is the standard version control program in the development world today. It is often integrated with development editors.

The core component of Git is the Git repository. This is a storage area where versions of code and related files are stored. Version control is managed within this local directory. The repository may be stored on a single developer's workstation, or this repository may be centrally stored and then cloned to the developer's workstation. Organizations may choose to have a centralized Git repository on premise, or to use an online solution like GitHub. A centralized repository is not required, however.

To get started with Git, use an available package manager to install the **git** package.

### THE git COMMAND

The **git** command is used to manage Git repositories. Using **git** you can create a repository, add files to the repository, commit changes to the repository, pull down files from another repository, and much more. You can perform these tasks by issuing various subcommands within the larger **git** command.

### SYNTAX

The syntax of the **git** command is **git [options] {subcommand}**

### git SUBCOMMANDS

The following are some of the major subcommands that you can use with the **git** command.

| Subcommand    | Used To                                                                         |
|---------------|---------------------------------------------------------------------------------|
| <b>config</b> | Set options for a repository or for Git users, as well as other global options. |
| <b>init</b>   | Create a Git repository or reinitialize an existing one.                        |
| <b>clone</b>  | Create a working copy of an existing repository.                                |
| <b>add</b>    | Add files to be tracked by the Git repository.                                  |

| Subcommand | Used To                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------|
| commit     | Update the Git repository with your changes, creating a "snapshot" of that repository.                |
| status     | Display the status of the repository.                                                                 |
| branch     | Manage branches, or pointers to specific repository snapshots after committing changes.               |
| merge      | Integrate changes from one branch into a "master" branch.                                             |
| pull       | Acquire and merge changes made to other repositories and branches into the local working copy.        |
| push       | Upload a local working copy of a repository to a remote repository, such as a centralized repository. |
| log        | Display the changes made to a local repository.                                                       |
| checkout   | Switch to a specific branch to work with.                                                             |

## EXAMPLE GIT PROCESS FOR LOCAL REPOSITORIES

The following is an example process flow for creating a local repository and committing changes to it:

1. Configure global settings, including a user name, email address, etc.:
  - a. `git config --global user.name 'User'`
  - b. `git config --global user.email 'user@domain.tld'`
2. Create a directory where your project will reside. Change into that directory, and then initialize it with Git to designate it as a Git repository:
  - a. `mkdir /dev-project`
  - b. `git init /dev-project`
3. Add project files to the repository. These are the files that make up the actual development project you are storing and controlling with Git. Just as you have seen with Linux configuration files, in development projects you want to work with copies of the project, not the original file itself. That makes it far easier to roll back to the original project in the event of a mistake. In Git, you create a working copy by using the `clone` subcommand:
  - `git clone /dev-project`
4. Add project files to the Git tracking system:
  - `git add myfile`
5. Commit the changes to take a snapshot of your project. At this stage, you can also enter a message that summarizes what changes you made. Make sure these messages are clear:
  - `git commit -m 'Initial commit'`
6. Retrieve the current status of changed files. If three files were being worked on for a particular step in the project, but only two were ready to be committed at this time, they would show up here as "added" but not yet committed. The commit process could be executed once edits to all three files are complete:
  - `git status`

## BRANCHING

Optionally, you can work with Git branches:

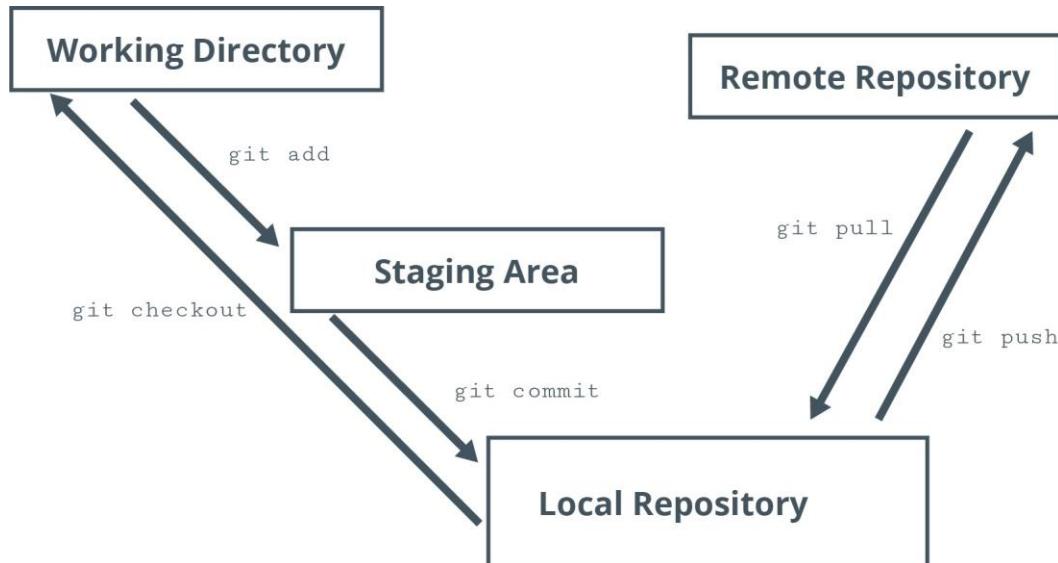
1. Create a branch of the master copy of the code:

- `git branch newbranch`
2. Make changes, and then integrate (merge) those changes back into the master branch. This integrates the changes, creating a new-and-improved version of the original. At this point, the branch that was being used to create the changes can be removed. The changes are now in the master branch:
- `git merge newbranch`

## EXAMPLE GIT PROCESS FOR COLLABORATION

The following is an example process flow for collaborating with other developers using Git:

1. Pull other developers' proposed changes and merge them into the local repository (the local working copy):
  - `git pull otherbranch`
2. Push your own changes to a remote repository. A development environment will usually include a central repository, perhaps in the cloud, and each developer has their own local copy of the repository. Changes made locally can then be uploaded to the central repository using the `Git push` command:
  - `git push <remote repository> mybranch`
3. See what changes were merged and what other actions were taken on the repository. Run the `git log` command inside the project directory. For example, enter `git log --since=10.days` to see all of the commits in the last 10 days. You can use this information to troubleshoot any issues introduced in configuration files, scripts, or any other data tracked by the repository.
4. Navigate or switch between branches of a project by using the `Git checkout` command. This enables developers to focus their attention on different branches of the same project:
  - `git checkout specificbranch`



An example Git process flow.

## GIT FILES

The `.gitignore` file exists within the repository. The purpose of the file is to identify files that should be ignored during a commit action. For example, you may have a `README.txt` file or a project `To-Do.txt` list that does not need to be committed as part of the project, but does reside in the project directory for convenience. These files are identified in the project's `.gitignore` file.

The `*.git/` directory contains all the files Git uses to manage version control for your project. It is a single location where Git stores all of its information. The directory resides in the project directory and is created with the `git init` command.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 14-4

## Discussing Version Control Using Git

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. **You need to set up a new Git repository for a group of developers. Which command begins the process of setting up a new Git repository?**
  
2. **Your developer group wants to implement Git so that they can manage code versioning. They ask you to install it on the dev01 CentOS server. How can you install Git?**
  
3. **You have created a new file, statistics.txt, and want to add it to the Git repository. Which command can you execute to add the file?**
  
4. **Over the past week, you have made several changes to the markets.txt file but have not committed those changes. How can you commit the changes?**
  
5. **Having returned from vacation today, you want to check the progress of the new coding project your group is working on. Which command can you issue to check the revision history of the repository?**

# Activity 14-5

## Implementing Version Control Using Git

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account.

### SCENARIO

The development team needs a way to easily manage the different versions of the code they write. Multiple developers will be working in conjunction on the same project, so they need to a way to minimize conflicts while being able to revert to older versions of code, if necessary. So, you'll set up a Git repository for the developers so that they have a distributed version control system to work from.

1. Install and configure a Git repository.
  - a) Enter the following and then wait for the installation process to complete:  
`sudo yum -y install git --disablerepo=internal-repo`
  - b) Enter `git config --global user.name 'Student User'`
  - c) Enter the following:  
`git config --global user.email 'student##@develetech.com'`
  - d) Create a directory in your home directory called `dev-project` and use the `CD` command to enter the directory.
  - e) Enter `git init` to designate the `dev-project` directory as a Git repository. A message is returned from Git indicating the repository is initialized.
  - f) Enter `ls -a` to view the `.git` directory created by the initialization process.
2. Create and manage a project using Git.
  - a) Use a text editor to create a file named `HelloWorld.txt`
  - b) Enter the following text in the `HelloWorld.txt` file:  
`Hello, World! From Student##`
  - c) Save your changes and close the editor.
  - d) Enter `git status` to check the status of the `HelloWorld.txt` file.  
 The file is marked as "Untracked", meaning it is not yet managed by Git.
  - e) Enter `git add HelloWorld.txt` to enable Git to manage the file.
  - f) Enter `git commit -m "Initial Commit"`  
 This updates Git with the version information for the `HelloWorld.txt` file.
  - g) Enter `git status` to check the status of the `HelloWorld.txt` file.  
 The output indicates that there is nothing to commit because the `HelloWorld.txt` file version is now managed by Git.
3. Commit a change to the Git repository.

- a) Use a text editor to open the **HelloWorld.txt** document, and add the following on a new line:  
**Git version control test**
- b) Save your changes and close the editor.
- c) Enter **git status** and observe that Git reports the **HelloWorld.txt** file as modified, but that the changes are not yet committed to the repository.
- d) Enter **git add HelloWorld.txt** to stage the changes.
- e) Enter **git commit -m "Revision 1"** to commit the changes to the master copy of the file.
- f) Enter **git status** and notice that there are now no changes to commit to the repository.
- g) Enter **git log** to view the revision history of the repository.

Times and dates for the initial commit and the revision have been logged.

---

# Topic C

## Identify Orchestration Concepts



### EXAM OBJECTIVES COVERED

5.3 Summarize orchestration processes and concepts.

So far, you've implemented automation on a relatively small scale. However, you may be in charge of hundreds, or even thousands, of systems in a large corporate infrastructure. Automation alone is not enough to manage the deployment and maintenance of all of these systems. This is where orchestration comes into play.

### ORCHESTRATION

**Orchestration** enables the automation of multiple related tasks—an entire workflow. One example of orchestration might be the deployment of a web app. The deployment may include the installation and configuration of the web server, the installation and configuration of a MySQL™ database server, and the installation and configuration of an application server, as well as all supporting software. Orchestration would manage each of the steps involved, even though there may be different operating systems and configuration requirements involved. Orchestration is used in both on-premise and cloud-based solutions.

### INFRASTRUCTURE AUTOMATION

Automation is the process of accomplishing a configuration task without human intervention. This is different than orchestration. Automation refers to a single task, whereas orchestration manages a larger scale series of tasks. For example, one system administrator might automate the installation of the Python® package in a Linux deployment, while another administrator might orchestrate the setup of a combined web and database server with all necessary configurations. Orchestration may be thought of as a series of automation tasks to accomplish a large-scale deployment of applications, virtual machines, or entire inter-related infrastructures.

### BUILD AUTOMATION

Build automation specifically emphasizes the initial operating system deployment. One example of build automation is the use of Kickstart files with Red Hat-derived distributions. These files can be referenced by the installation procedure and the operating system is then deployed according to the instructions in the Kickstart file.

### INFRASTRUCTURE AS CODE

**Infrastructure as code** is a name for orchestration tools that manage the entire deployment and configuration process through scripting and code files, rather than through traditional software tools. Infrastructure as code relies on a single configuration specification to deploy the supporting infrastructure (the operating system) and the necessary applications.

## ORCHESTRATION TOOLS

The following table lists some common tools used to implement orchestration.

| Tool       | Description                                                                                                                                                                                                                                   |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ansible    | Ansible uses YAML files to create repeatable "playbooks" that define a desired configuration state. Ansible is an agentless solution that delivers files over SSH connections. Red Hat emphasizes the use of Ansible.                         |
| Puppet®    | Puppet uses manifest files (written in Ruby) to define infrastructure as code for application, cloud, and infrastructure orchestration. Puppet uses an agent on the target nodes.                                                             |
| Chef™      | Chef uses "cookbooks" to deliver configuration declarations to cloud and on-premises managed systems.                                                                                                                                         |
| Kubernetes | Kubernetes is an open source solution that provides container deployment and application orchestration for cloud and on-premises container environments. You define a desired state and Kubernetes configures containers to match that state. |
| OpenStack® | OpenStack was originally a joint Rackspace and NASA project, usually deployed as an IaaS solution to manage cloud resources. OpenStack can orchestrate the deployment of a Linux, Apache, MySQL, PHP (LAMP) service, for example.             |

## AGENT-BASED VS. AGENTLESS ORCHESTRATION

Agent-based orchestration tools require that a software component reside on the managed device. Agentless tools do not require additional software to exist ahead of time on the managed system.

## PROCEDURES

Orchestration procedures will vary by which solution is used, but in general, orchestration steps involve defining a desired configuration and then delivering that configuration file to the destination system. The configuration definition is then processed, setting the system's configuration to match the definition file.

The following is an example of Chef orchestration procedures:

1. Administrators configure and test a Chef "cookbook" of configurations.
2. The cookbook is delivered to the specified destination client systems.
3. The Chef client processes the cookbook, configuring the node appropriately.

## ATTRIBUTES

Orchestration attributes define tasks to be managed by the orchestration process. Administrators can use these attributes to identify specific configurations that need to be set by the orchestration process. OpenStack orchestration relies on attributes, for example.

## INVENTORY MANAGEMENT

Inventory management of hardware, virtual machines, operating systems, applications, and configurations may all be managed through orchestration tools. Different tools offer different features for inventory management, including reporting. Inventory is crucial because administrators cannot manage what they don't know about.

## AUTOMATED CONFIGURATION MANAGEMENT

The benefits of configuration management include consistently configured systems and a more efficient build process. Ensuring that all systems meet a given configuration baseline helps to enforce security requirements and service-level agreements, and makes change management more efficient.

# Activity 14-6

## Identifying Orchestration Concepts

### SCENARIO

Develetech is expanding into web hosting services. You have been asked to research how orchestration could be used to make web hosting more efficient and cost- effective.

1. **What is the difference between automation and orchestration? Why does this difference matter to Develetech?**
2. **Give examples of open source orchestration tools used in enterprises.**
3. **Explain the three general steps in orchestration.**
4. **Ansible is a very popular orchestration tool in both large and small company networks. How does Ansible deliver configuration files to nodes?**
5. **Develetech will be providing LAMP (Linux, Apache, MySQL, PHP) web hosting packages for customers. How might orchestration benefit Develetech in this context?**
6. **Can orchestration help Develetech with cloud-based web hosting? Why or why not?**

## Summary

In this lesson, you automated your day-to-day administrative tasks using techniques like cron jobs and version control systems. By doing this, you can improve productivity and minimize the errors that are common in manual administration.

**Do you, or would you, consider using a version control system like Git? Why or why not?**

**What kind of cron jobs might you create on your Linux systems?**



**Practice Question:** Additional practice questions are available on the course website.

# Lesson 15

## Installing Linux

**LESSON TIME: 1 HOUR, 30 MINUTES**

### LESSON INTRODUCTION

In many cases, the Linux® systems you'll work on will have already been built for you. But sometimes, you'll need to build a system from scratch. You must be able to install Linux in a way that meets your organization's business needs, whatever they may be. So, you'll wrap things up by configuring a Linux installation to your desired specifications.

### LESSON OBJECTIVES

In this lesson, you will:

- Prepare to install the Linux operating system.
- Perform the installation of Linux.

# Topic A

## Prepare for Linux Installation

Before you actually begin installation, it's important that you take some steps to prepare. You'll need to do some research and gather the appropriate information that will help guide you during the installation process.

### SYSTEM REQUIREMENTS

It's important to think of system requirements as more than just what hardware components are necessary to get a system up and running—they also specify what is necessary to keep that system operational and able to perform its assigned function. So, you may be able to install Linux on a computer just fine, but that doesn't mean it will perform optimally.

The system requirements for a Linux system will vary greatly based on a number of different factors:

- The Linux distribution you're using. Different distros have different recommendations and baseline requirements as far as CPU speed, RAM, storage space, and more. Some distros are specifically designed to be lightweight; i.e., they consume much fewer resources than a standard enterprise or consumer distro.
- Hardware compatibility. Even if a CPU, GPU, etc., is more than powerful enough to run a Linux distro, that doesn't mean the distro (or the Linux kernel) supports it. You need to choose hardware that has available, stable drivers
- The general category of the system. Is the system a server? A workstation? Something else? Each category lends itself to different types of components. For example, CPUs like the Intel® Xeon® are designed for servers, as they place emphasis on error correction and reliability, as well as having many cores per processor. A desktop CPU, on the other hand—like an Intel® Core™ i7—is designed for single-user scenarios and is typically less powerful than a server CPU.
- The intended function of the system. Beyond whether or not the system is a server, workstation, etc., what role does it actually play in the organization? Is the server a web server? A database server? Does the workstation need to be used for graphic design? Does it need to be used for word processing? Ultimately, the more demanding an application is, the more resources you'll need to give it. A web server might need to handle thousands of network connections per day, and should therefore be running on adequate networking infrastructure.
- The specific applications that need to run on the system. You might be able to estimate that a workstation used for graphic design needs a discrete video card. However, you might not know the exact video card to choose until you know what application the user will be working with. Some software works best with certain hardware.
- Price. No organization's budget is limitless, and you should expect to not always be able to purchase the most powerful hardware. You need to choose solutions that are the right fit for your organization, while at the same time within your assigned budget. Prioritizing resources is a crucial skill in helping you make these decisions.

## PARTITIONING STRATEGY

Prior to the installation process, you must be careful to plan an appropriate partitioning strategy. Servers have different storage needs than individual workstations. If the partition where the root of the file system is mounted fills up, the system will crash. User home directories and log files are both examples of directories that can unexpectedly increase in size very rapidly. By isolating these directories to their partitions, you can prevent a server crash.

Linux usually uses a dedicated partition for virtual memory storage (i.e., swap space). This partition is unusable for any other kind of storage and has its own file system. It is important to plan for the swap partition at the same time you plan for data storage partitions. A general guideline is that the size of the swap partition should be two times the quantity of RAM, though this number will vary.

## PARTITIONING EXAMPLES

The following is a general partitioning strategy for a Linux server:

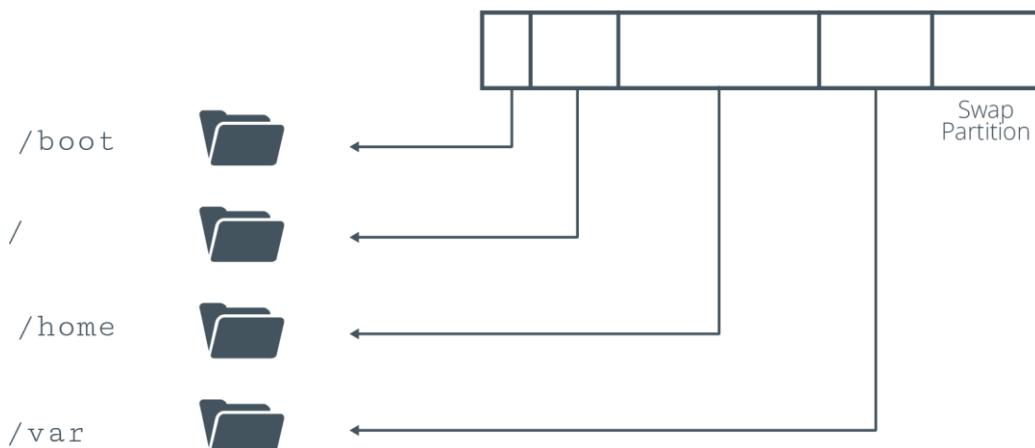
- Single partition as `/` (the root file system).
- Single partition as `/home` to isolate user home directories from the rest of the storage space.
- Single partition as `/var` to isolate log files from the rest of the storage space.
- Single partition dedicated to swap space, typically twice the quantity of RAM.

And a general partitioning strategy for Linux workstations:

- Single partition as `/` (the root file system).
- Single partition dedicated to swap space, typically twice the quantity of RAM.



**Note:** Individual Linux workstations are not usually shared with other users and often have very high capacity storage drives, so the risk of a single user unexpectedly filling an entire drive is significantly less. Furthermore, individual workstations are usually less important to the business than Linux servers.



An example partitioning strategy.

## HARDWARE COMPATIBILITY

The first thing you should do before purchasing any hardware component is to check whether it is compatible with the relevant Linux distribution. Some distros maintain a **hardware compatibility list (HCL)** to help you with this effort. An HCL is a database that stores the vendors and models of all hardware devices that a distro supports in some capacity. Each distro may maintain its own HCL, which is usually published

online. There is no standard format for an HCL, and many distros don't even have an authoritative HCL.



**Note:** For a generic Linux hardware compatibility list, visit <https://www.linuxquestions.org/hcl/>.

So, before you install Linux, you should gather hardware information about your system. Much of this information is available in your system documentation, whether it's the printed manual that came with a component or its documentation you can find on the manufacturer's website. You can also gather hardware device data from a low-level interface like BIOS/UEFI. Failing either of those options, you may need to open the hood and look inside. Many components have vendor and model information printed on them.

## QUESTIONS TO ADDRESS WHEN CHOOSING HARDWARE

Some of the questions that you should address before purchasing a hardware component are listed in the following table. Note that the questions "Who is the manufacturer?", "What is the model?", and "Is there driver support?" apply to pretty much all component types.

| Component                            | Questions to Address                                                                                                                                                                                                                                                                                                                                     |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Storage drive                        | <ul style="list-style-type: none"> <li>• How many devices are installed?</li> <li>• What technology does the interface controller use (SATA, SCSI, NVMe, etc.)?</li> <li>• What is the drive's capacity?</li> <li>• Is the drive mechanical or does it use solid-state technology?</li> <li>• What read/write speeds is the drive capable of?</li> </ul> |
| CPU                                  | <ul style="list-style-type: none"> <li>• Does it target the server or desktop market?</li> <li>• What is the clock rate?</li> <li>• What is the cache size?</li> <li>• How many physical cores are there?</li> <li>• Does it support hyper-threading?</li> </ul>                                                                                         |
| Memory                               | <ul style="list-style-type: none"> <li>• How much RAM is installed?</li> <li>• Are there enough DIMM slots on the motherboard to accommodate the necessary RAM modules?</li> <li>• What is the generation of DDR SDRAM (DDR3, DDR4, DDR5, etc.)?</li> </ul>                                                                                              |
| Network card                         | <ul style="list-style-type: none"> <li>• What networking technologies does it support (Ethernet, Wi-Fi, etc.)?</li> <li>• What is the maximum bandwidth supported?</li> </ul>                                                                                                                                                                            |
| Input device (mouse, keyboard, etc.) | <ul style="list-style-type: none"> <li>• Does it support a hotpluggable interface?</li> <li>• What version of USB does it support, if any?</li> </ul>                                                                                                                                                                                                    |
| Monitor                              | <ul style="list-style-type: none"> <li>• What connection interfaces does it support?</li> <li>• What is the refresh rate range of the monitor?</li> <li>• What is the maximum supported pixel resolution of the monitor?</li> </ul>                                                                                                                      |

| Component       | Questions to Address                                                                                                                                                                                                                                                 |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Display adapter | <ul style="list-style-type: none"> <li>• What chipset is used?</li> <li>• How much video RAM does it use?</li> <li>• Is it integrated into the CPU or motherboard, or is it a discrete video card?</li> <li>• What connection interfaces does it support?</li> </ul> |

## INSTALLATION METHODS

In most cases, installing an operating system will require you to boot into a special installation environment. Therefore, the different boot methods you learned about earlier can also be used to boot installation media, including:

- Booting from removable installation media like DVDs and USB thumb drives. This method typically requires you to "burn" the installation files onto the DVD, or use a program to prepare the files on the USB drive. Adding the files to the USB drive normally, like you would when storing data, won't make it bootable. Installation files are often packaged in a system image format like ISO or IMG. By default, most BIOS/UEFI environments enable booting from removable media, but you may need to enable this manually on some systems.
- Booting from media installed on the local drive. For example, you can download an ISO of a different distro of Linux onto your existing Linux system. You can then configure GRUB 2 to boot from this ISO, as long as it is placed on a partition that doesn't currently contain a bootable OS.
- Boot from media that is delivered over a network. You can use protocols like PXE and NFS to service installation media to a client, who can use that media to boot into the installation environment.



The different media used to install Linux.

## GUIDE

Use the following guidelines when preparing to install Linux.

## PREPARE TO INSTALL LINUX

When preparing to install Linux:

- Consider the different factors that go into identifying system requirements, such as cost, system role, and Linux distribution to install.
- Consider partition strategies based on the role of the Linux system. Linux servers may play many roles and the partition strategies will vary depending on those roles. Linux workstations do not usually require a complex partition strategy.
- Check an HCL or other online resource to see if your hardware is supported by the relevant distro.
- Address various questions pertaining to each hardware component before purchasing them.
- Choose an installation method that is most efficient for you and your organization.

# Activity 15-1

## Preparing to Install Linux

### SCENARIO

As a system administrator for Develetech, you need to create systems for different purposes as part of a general migration to Linux. Those systems will be:

- A public-facing web server that will receive tens of thousands of page views per day.
- A desktop Linux implementation for the graphic designer, who will use it to develop illustrations and other graphic assets, and for testing web pages.
- A system used for testing software before it is deployed throughout the organization.

The hardware systems that are available to you at this time include:

- **System A:** A newly built, custom system with 32 GB RAM, quad-core Intel Core i7 processor, 2 TB hard drive, Fast Ethernet (100 Mb/s) network card, and discrete video card with HDMI and DisplayPort connectors.
- **System B:** A decommissioned system with 8 GB RAM, 18-core Xeon processor, 500 GB SSD, Gigabit Ethernet (1 Gb/s) network card, and onboard graphics adapter with a DVI connector.
- **System C:** A laptop with 4 GB RAM, dual-core AMD A6 processor, 250 GB hard drive, Fast Ethernet (100 Mb/s) network card, and onboard graphics adapter with an HDMI connector.

---

**1. Which system would you set up for each purpose, and why?**

**2. As far as the hardware, what else do you need to consider before you can install Linux on these systems?**

- 3. As part of the general roll-out at Develetech, what questions would you need to ask each user to be sure that the new system will meet their needs?**
  
  - 4. Although there are multiple methods of installing Linux onto a system, which is the most common and the likely choice for most users?**
  
  - 5. Assuming you want to use a USB thumb drive to install Linux on these systems, what do you need to consider before doing so?**
  
  - 6. During installation, why is it important to separate certain directories onto their own file systems?**
  
  - 7. For Linux servers, which directories should typically be separated onto their own file systems?**

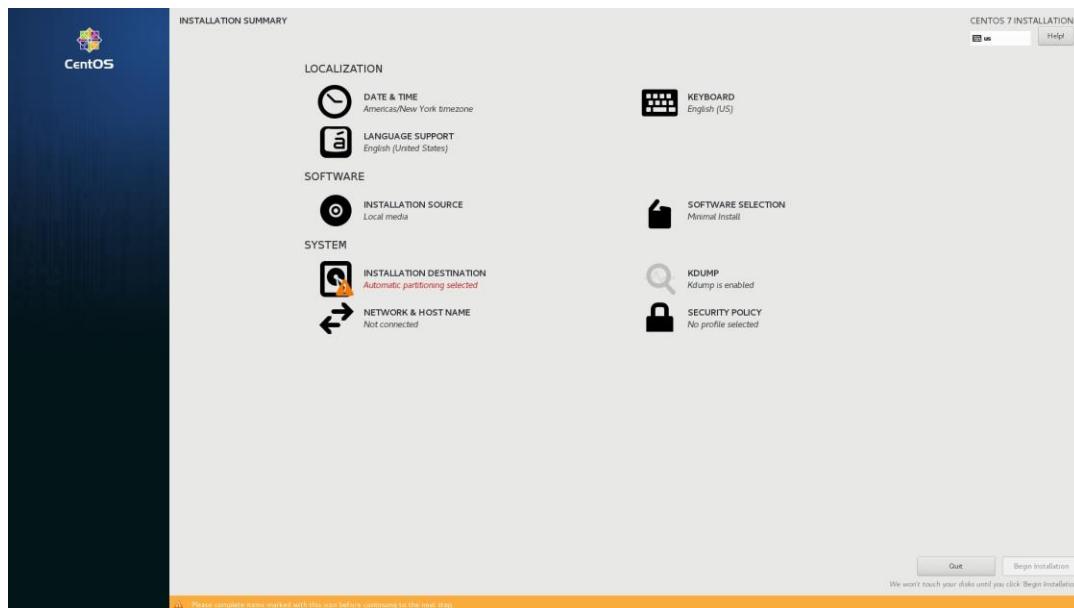
# Topic B

## Perform the Installation

Now that you've prepared to the best of your ability, the time has come to install Linux.

### MANUAL INSTALLATION STEPS

Although you can automate the installation and deployment of a Linux system—including as part of a larger orchestration effort—there may be times when you need to perform a step-by-step manual installation. The actual steps will differ widely depending on the distro you're installing, as each will have its own installer program. However, most installer programs feature a GUI and enable you to configure some common settings to install Linux with.



*Installing CentOS 7.*

General installation steps can include, but are not limited to:

- Configure the system language and keyboard layout to use. This step almost always comes first to set a baseline for understanding the rest of the installation process.
- Configure the time zone and other date and time settings for the system to use. There may be an option to synchronize with a Network Time Protocol (NTP) server.
- Choose the installation media to use. Even though you booted into the installer using installation media, some installers enable you to choose a different source to continue with the installation. For example, if you started the installer with an ISO prepared on a USB thumb drive, you can configure the installation to use a network repository instead.
- Choose which software environment and components to install. Many distros offer a choice of one of several base environments. A base environment might be a minimal install without a GUI and with few user space programs; a full-fledged GUI install with all available programs; and everything in between. Depending on the distro, you may be able to get more granular and choose specific software to install.

- Partition and configure logical volumes on one or more available storage drives. You can design the structure of the root file system here (its size, file system type, etc.), as well as any other file systems to add on installation.
- Configure the system's networking identity. You can configure one or more available interfaces with IP addresses, hostnames, and other networking information.
- Configure user accounts. Typically you'll configure the root user's password, and/or a specific administrator account or other standard users.
- Configure security policies. Some installers enable you to apply security profiles and policies to the system upon installation.



**Note:** To learn more, check the Video tab on the course website for any videos that supplement the content for this lesson.

# Activity 15-2

## Discussing Installing Linux

### SCENARIO

Answer the following questions to check your understanding of the topic.

1. **Depending on the Linux distribution you have chosen to install, which accounts are you prompted to set up during installation?**
2. **During installation, you are prompted to set up a system's networking identity. What information is included in a system's networking identity?**
3. **On larger networks, installing Linux manually each time is not efficient. How do larger organizations install Linux, if not manually?**
4. **You want to manually install Linux on a new laptop that has no DVD drive, and you do not have access to one. How can you install Linux onto the laptop?**
5. **What are usually the first two pieces of information you are prompted to provide during a Linux installation?**

# Activity 15-3

## Installing Linux

### BEFORE YOU BEGIN

You are logged in to the GUI as your student account. Previously, you created and saved a virtual machine with a CentOS 7 installation image.

### SCENARIO

Now that your preparations are complete, you're ready to install Linux on the various systems you selected. You'll start by installing CentOS 7 on the VM you created earlier. As you go through the installation, you'll configure various options so that the base environment will be automatically set up to your specifications.

1. Load the previously created VM.
  - a) At a terminal, enter **`sudo virsh restore saved-vm`**  
This restores the VM you created earlier from its saved state.
  - b) From the desktop menu, select **Applications**→**System Tools**→**Virtual Machine Manager**.
  - c) Enter the root password.
  - d) Right-click the **devtech-install** VM and select **Open**.
  - e) Wait for the installation media to finish its check.  
You can press **Esc** to skip the check, but it's wise to check the media at least once when setting up production systems.
  
2. Start by configuring localization settings.
  - a) If necessary, expand the virtual machine window so it's easier to see.



**Note:** You can also select the **Switch to fullscreen view** button.

  - b) On the **WELCOME TO CENTOS 7** page, select **Continue** to accept the default language settings.
  - c) On the **INSTALLATION SUMMARY** page, under the **LOCALIZATION** section, select **DATE & TIME**.
  - d) Select your time zone, then select **Done**.
  
3. Select the software components and base environment to install.
  - a) Under the **SOFTWARE** section, select **SOFTWARE SELECTION**.
  - b) From the **Base Environment** list, select **Server with GUI**.
  - c) From the **Add-Ons for Selected Environment** list, check the **KDE** check box.  
By default, the **Server with GUI** selection will install most tools necessary for the configuration and maintenance of general server infrastructure, along with GNOME as the default GUI. You're also installing KDE alongside that for users to have a choice of desktop environment.
  - d) Select **Done**.

4. Wipe the storage device to start fresh.
  - a) Under the **SYSTEM** section, select **INSTALLATION DESTINATION**.
  - b) On the **Device Selection** page, observe the **Virtio Block Device**.  
This is the 12 GB virtual storage device that was created when you first installed the VM.
  - c) Under **Other Storage Options**, ensure **Automatically configure partitioning** is selected.
  - d) Check the **I would like to make additional space available** check box.
  - e) Select **Done**.
  - f) In the **RECLAIM DISK SPACE** dialog box, verify that the **vda** device is selected, and that it has 12 GB of free space.  
This is because the virtual storage device you created is currently empty. Still, it's useful to practice wiping a storage device in order to start fresh.
  - g) Select **Delete all**.
  - h) Select **Reclaim space**.

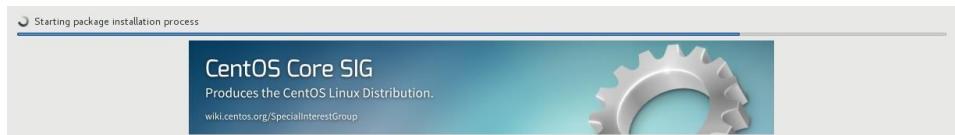
5. Configure the partitioning scheme to use.
  - a) On the **INSTALLATION SUMMARY** page, select **INSTALLATION DESTINATION** again.
  - b) Under **Other Storage Options**, select **I will configure partitioning**.
  - c) Select **Done**.
  - d) Under **New CentOS 7 Installation**, verify that no mount points have been created yet, and that the default partitioning scheme will use LVM.
  - e) Select **Click here to create them automatically**.
  - f) Verify that three partitions/volumes were created: **/boot**, **/** (root), and **swap**.  
Notice that there is no separate **/home** volume. This is because the CentOS 7 installer only creates a separate **/home** volume by default when the storage device is 50 GB or more. In this case, the **/home** directory will be located within the root volume.
  - g) Select the **/boot** partition and note its default capacity, device type (partitioning scheme), and file system type.
  - h) Select the **/** (root) volume and the **swap** volume and note their defaults as well.  
These will be created as logical volumes within the **centos** volume group.
  - i) At the bottom-left of the page, note the total space of the storage device as well its available space.  
This reflects the intended partitioning scheme; no changes will be made to the drive until installation begins in earnest.
  - j) Select **Done**.
  - k) In the **SUMMARY OF CHANGES** dialog box, select **AcceptChanges**.

6. Configure networking.
  - a) On the **INSTALLATION SUMMARY** page, select **NETWORK & HOSTNAME**.
  - b) In the **Host name** text box at the bottom-left, type **devtech-vm##** where **##** refers to your student number, then select **Apply**.
  - c) In the list of devices on the left, verify that **Ethernet (eth0)** is selected.  
This is the virtual network interface that was created for the VM to use.
  - d) Select **Configure** at the bottom-right of the page.
  - e) In the **Editing eth0** dialog box, select the **IPv4 Settings** tab.
  - f) From the **Method** drop-down list, select **Manual**.
  - g) To the right of the **Addresses** list, select the **Add** button.
  - h) For the **Address**, type **10.50.1.2##** where **##** is your student number. For example, the IP address for student 01's VM would be **10.50.1.201**



**Note:** Your instructor might provide you with different addresses than those listed in these steps.

- i) Press **Tab**, then for the **Netmask**, type **255.255.255.0**
  - j) Press **Tab**, then for the **Gateway**, type **10.50.1.1** and press **Enter**.
  - k) In the **DNS Servers** text box, type **8.8.8.8**
  - l) Select **Save**.
  - m) Select the slider at the top-right to turn the interface **On**.
  - n) Verify that the interface details are as you expect, then select **Done**.
- 7.** Begin installation and configure user accounts.
- a) Select **Begin Installation**.
  - b) Observe the progress bar at the bottom, indicating that CentOS is in the process of being installed.



- c) Under **USER SETTINGS**, select **ROOT PASSWORD**.
- d) In the **Root Password** text box, type **Pa22w0rd**
- e) In the **Confirm** text box, type **Pa22w0rd**
- f) Select **Done**, then, at the bottom of the screen, verify that CentOS points out that this password is weak because it's based on a dictionary word.  
In a production environment, you'd want to choose a much stronger password.
- g) Select **Done** again to agree to use the password.
- h) Select **USER CREATION**.
- i) In the **User name** text box, type **student##** where **##** refers to your student number.
- j) Check the **Make this user administrator** check box.
- k) In the **Password** and **Confirm password** text boxes, type **Pa22w0rd**
- l) Select **Done** twice to confirm the password.  
In addition to creating a stronger password, you'd also want to make your user password different than the root password in a production environment.
- m) Wait for the system to finish installing.

**Note:** The remainder of the installation process may take up to 30 minutes.



- 8.** Complete the installation process.
- a) When installation finishes, select **Reboot**.
  - b) From the VM window, select **Virtual Machine→Run** to restart the VM.
  - c) On the **INITIAL SETUP** page, select **LICENSING INFORMATION**.
  - d) Check **I accept the license agreement** and select **Done**.
  - e) Select **FINISH CONFIGURATION**.
  - f) Verify that you are greeted with the sign in screen, indicating that CentOS 7 was successfully installed.
- 9.** Verify your new system's configurations.
- a) Sign in as your student account.
  - b) Using what you've learned, check the VM for the following:
    - Storage partition and logical volume configurations.
    - User accounts.
    - Networking configurations.
    - Connectivity with other classroom computers.
    - Internet connectivity.

- Additional software packages.
- c) When you're done, from the VM window, select **Virtual Machine**→**Shut Down**→**Shut Down**.
- d) Close the VM window.
-

## Summary

In this lesson, you prepared to install Linux by validating basic system requirements as well as requirements prompted by your organization's business needs. Then, you were able to successfully install the Linux operating system.

**What Linux distributions do you think you'll install in your organization, or have you installed in the past?**

**Do you anticipate any hardware compatibility issues in your installations? Why or why not?**



**Practice Question:** Additional practice questions are available on the course website.

# Course Follow-Up

Congratulations! You have completed *The Official CompTIA® Linux+® (Exam XK0-004)* course. You have acquired the essential skills and information you will need to configure, manage, operate, and troubleshoot a Linux environment by using security best practices, scripting, and automation.

You also covered the objectives that you will need to prepare for the CompTIA Linux+ (Exam XK0-004) certification examination. If you combine this class experience with review, private study, and hands-on experience, you will be well prepared to demonstrate your Linux administration expertise both through professional certification and solid technical competence on the job.

## What's Next?

Become a CompTIA Linux+ Certified Professional!

CompTIA Linux+ certified professionals have validated the competencies required of an early-career IT professional supporting Linux systems. CompTIA Linux+ is the engine to power success for IT pros looking to validate the Linux skills employers need and ensures they can get the most out of cloud, IoT, and cybersecurity investments.

The exam is the only job-focused Linux certification covering the most current and relevant subject matter as demanded by hiring managers around the world.

In order to become a CompTIA Linux+ Certified Professional, you must successfully pass the Linux+ (Exam Code XK0-004) exam.

In order to help you prepare for the exams, you may want to invest in CompTIA's exam prep product, CertMaster Practice for Linux+.

CertMaster Practice is an online knowledge assessment and certification training companion tool specifically designed for those who have completed *The Official CompTIA Linux+* course. It helps reinforce and test what you know and close knowledge gaps prior to taking the exam.

CertMaster Practice features:

- Adaptive knowledge assessments with feedback, covering all domains of the Linux+ exam.
- Practice tests with performance-based questions.
- Question-first design and smart refreshers to get feedback on the questions you get wrong.
- Learning analytics that track real-time knowledge gain and topic difficulty to help you learn intelligently.

## Taking the Exam

When you think you have learned and practiced the material sufficiently, you can book a time to take the test.

## Preparing for the Exam

We've tried to balance this course to reflect the percentages in the exam so that you have learned the appropriate level of detail about each topic to comfortably answer the exam questions. Read the following notes to find out what you need to do to register for the exam and get some tips on what to expect during the exam and how to prepare for it.

Questions in the exam are weighted by domain area as follows.

### CompTIA Linux+ (Exam XK0-004) Certification Domain Areas Weighting

|                                       |     |
|---------------------------------------|-----|
| 1.0 Hardware and System Configuration | 21% |
| 2.0 Systems Operation and Maintenance | 26% |

**CompTIA Linux+ (Exam XK0-004) Certification Domain Areas Weighting**

|                                           |     |
|-------------------------------------------|-----|
| 3.0 Security                              | 19% |
| 4.0 Linux Troubleshooting and Diagnostics | 20% |
| 5.0 Automation and Scripting              | 14% |

For more information about how to register for and take your exam, please visit the CompTIA website: <https://certification.comptia.org/testing>

# Mapping Course Content to CompTIA® Linux+®(Exam XK0-004)

Achieving CompTIA Linux+ certification requires candidates to pass exam XK0-004. This table describes where the exam objectives are covered in this course.

| Domain and Objective                                                         | Covered In |
|------------------------------------------------------------------------------|------------|
| <b>Domain 1.0 Hardware and System Configuration</b>                          |            |
| <b>1.1 Explain Linux boot process concepts.</b>                              |            |
| • Boot loaders                                                               | 7A, 7B     |
| • GRUB                                                                       | 7B         |
| • GRUB 2                                                                     | 7B         |
| • Boot options                                                               | 7A         |
| • UEFI/EFI                                                                   | 7A         |
| • PXE                                                                        | 7A         |
| • NFS                                                                        | 7A         |
| • Boot from ISO                                                              | 7A         |
| • Boot from HTTP/FTP                                                         | 7A         |
| • File locations                                                             | 7A, 7B     |
| • /etc/default/grub                                                          | 7B         |
| • /etc/grub2.cfg                                                             | 7B         |
| • /boot                                                                      | 7A         |
| • /boot/grub                                                                 | 7A         |
| • /boot/grub2                                                                | 7A         |
| • /boot/efi                                                                  | 7A         |
| • Boot modules and files                                                     | 7A, 7B     |
| • Commands                                                                   | 7A, 7B     |
| • mkinitrd                                                                   | 7A         |
| • dracut                                                                     | 7A         |
| • grub2-install                                                              | 7B         |
| • grub2-mkconfig                                                             | 7B         |
| • initramfs                                                                  | 7A         |
| • efi files                                                                  | 7A         |
| • vmlinuz                                                                    | 7A         |
| • vmlinux                                                                    | 7A         |
| • Kernel panic                                                               | 7A         |
| <b>1.2 Given a scenario, install, configure, and monitor kernel modules.</b> |            |
| • Commands                                                                   | 6B, 6C     |
| • lsmod                                                                      | 6B         |
| • insmod                                                                     | 6B         |
| • modprobe                                                                   | 6B         |
| • modinfo                                                                    | 6B         |
| • dmesg                                                                      | 6C         |
| • rmmod                                                                      | 6B         |
| • depmod                                                                     | 6B         |

| Domain and Objective                                                             | Covered In |
|----------------------------------------------------------------------------------|------------|
| • Locations                                                                      | 6B         |
| • /usr/lib/modules/[kernelversion]                                               | 6B         |
| • /usr/lib/modules                                                               | 6B         |
| • /etc/modprobe.conf                                                             | 6B         |
| • /etc/modprobe.d                                                                | 6B         |
| <b>1.3 Given a scenario, configure and verify network connection parameters.</b> |            |
| • Diagnostic tools                                                               | 10C, 10F   |
| • ping                                                                           | 10F        |
| • netstat                                                                        | 10F        |
| • nslookup                                                                       | 10F        |
| • dig                                                                            | 10F        |
| • host                                                                           | 10F        |
| • route                                                                          | 10F        |
| • ip                                                                             | 10F        |
| • ethtool                                                                        | 10C        |
| • ss                                                                             | 10F        |
| • iwconfig                                                                       | 10C        |
| • nmcli                                                                          | 10C        |
| • brctl                                                                          | 10C        |
| • nmtui                                                                          | 10C        |
| • Configuration files                                                            | 10C, 10D   |
| • /etc/sysconfig/network-scripts/                                                | 10C        |
| • /etc/sysconfig/network                                                         | 10C        |
| • /etc/hosts                                                                     | 10D        |
| • /etc/network                                                                   | 10C        |
| • /etc/nsswitch.conf                                                             | 10D        |
| • /etc/resolv.conf                                                               | 10D        |
| • /etc/netplan                                                                   | 10C        |
| • /etc/sysctl.conf                                                               | 6B         |
| • /etc/dhcp/dhclient.conf                                                        | 10D        |
| • Bonding                                                                        | 10C        |
| • Aggregation                                                                    | 10C        |
| • Active/passive                                                                 | 10C        |
| • Load balancing                                                                 | 10C        |
| <b>1.4 Given a scenario, manage storage in a Linux environment.</b>              |            |
| • Basic partitions                                                               | 7A         |
| • Raw devices                                                                    | 7A         |
| • GPT                                                                            | 7A         |
| • MBR                                                                            | 7A         |
| • File system hierarchy                                                          | 4A, 4E     |
| • Real file systems                                                              | 4A         |
| • Virtual file systems                                                           | 4A         |

| Domain and Objective | Covered In         |
|----------------------|--------------------|
| • Relative paths     | 4E                 |
| • Absolute paths     | 4E                 |
| • Device mapper      | 4B                 |
| • LVM                | 4B                 |
| • mdadm              | 4B                 |
| • Multipath          | 4B                 |
| • Tools              | 4A, 4B, 4C, 4D, 4F |
| • XFS tools          | 4D                 |
| • LVM tools          | 4B                 |
| • EXT tools          | 4D                 |
| • Commands           | 4A, 4B, 4C, 4D, 4F |
| •   mdadm            | 4B                 |
| •   fdisk            | 4A                 |
| •   parted           | 4A                 |
| •   mkfs             | 4A                 |
| •   iostat           | 4F                 |
| •   df               | 4F                 |
| •   du               | 4F                 |
| •   mount            | 4C                 |
| •   umount           | 4C                 |
| •   lsblk            | 4D                 |
| •   blkid            | 4D                 |
| •   dumpe2fs         | 4D                 |
| •   resize2fs        | 4D                 |
| •   fsck             | 4D                 |
| •   tune2fs          | 4D                 |
| •   e2label          | 4A                 |
| • Location           | 4A, 4B, 4D, 4E     |
| •   /etc/fstab       | 4A                 |
| •   /etc/crypttab    | 4A                 |
| •   /dev/mapper      | 4A                 |
| •   /dev/disk/by-    | 4A                 |
| •     multipath      | 4A                 |
| •   /etc/mtab        | 4D                 |
| •   /sys/block       | 4E                 |
| •   /proc/partitions | 4D                 |
| •   /proc/mounts     | 4D                 |
| • File system types  | 4A                 |
| •   ext3             | 4A                 |
| •   ext4             | 4A                 |

| Domain and Objective                                                                | Covered In |
|-------------------------------------------------------------------------------------|------------|
| • xfs                                                                               | 4A         |
| • nfs                                                                               | 4A         |
| • smb                                                                               | 4A         |
| • cifs                                                                              | 4A         |
| • ntfs                                                                              | 4A         |
| <b>1.5 Compare and contrast cloud and virtualization concepts and technologies.</b> |            |
| • Templates                                                                         | 10E        |
| • VM                                                                                | 10E        |
| • OVA                                                                               | 10E        |
| • OVF                                                                               | 10E        |
| • JSON                                                                              | 10E        |
| • YAML                                                                              | 10E        |
| • Container images                                                                  | 10E        |
| • Bootstrapping                                                                     | 10E        |
| • Cloud-init                                                                        | 10E        |
| • Anaconda                                                                          | 10E        |
| • Kickstart                                                                         | 10E        |
| • Storage                                                                           | 10E        |
| • Thin vs. thick provisioning                                                       | 10E        |
| • Persistent volumes                                                                | 10E        |
| • Blob                                                                              | 10E        |
| • Block                                                                             | 10E        |
| • Network considerations                                                            | 10E        |
| • Bridging                                                                          | 10E        |
| • Overlay networks                                                                  | 10E        |
| • NAT                                                                               | 10E        |
| • Local                                                                             | 10E        |
| • Dual-homed                                                                        | 10E        |
| • Types of hypervisors                                                              | 10E        |
| • Tools                                                                             | 10E        |
| • libvirt                                                                           | 10E        |
| • virsh                                                                             | 10E        |
| • vmm                                                                               | 10E        |
| <b>1.6 Given a scenario, configure localization options.</b>                        |            |
| • File locations                                                                    | 8A         |
| • /etc/timezone                                                                     | 8A         |
| • /usr/share/zoneinfo                                                               | 8A         |
| • Commands                                                                          | 8A         |
| • localectl                                                                         | 8A         |
| • timedatectl                                                                       | 8A         |
| • date                                                                              | 8A         |
| • hwclock                                                                           | 8A         |
| • Environment variables                                                             | 13A        |

| Domain and Objective | Covered In |
|----------------------|------------|
| • LC_*               | 13A        |
| • LC_ALL             | 13A        |
| • LANG               | 13A        |
| • TZ                 | 13A        |
| • Character sets     | 8A         |
| •   UTF-8            | 8A         |
| •   ASCII            | 8A         |
| •   Unicode          | 8A         |

## Domain 2.0 Systems Operation and Maintenance

### 2.1 Given a scenario, conduct software installations, configurations, updates, and removals.

|                        |               |
|------------------------|---------------|
| • Package types        | 11A, 11E      |
| •   .rpm               | 11A           |
| •   .deb               | 11A           |
| •   .tar               | 11E           |
| •   .tgz               | 11E           |
| •   .gz                | 11E           |
| • Installation tools   | 11A, 11B, 11C |
| •   RPM                | 11A, 11B      |
| •   dpkg               | 11A, 11C      |
| •   APT                | 11A, 11C      |
| •   YUM                | 11A, 11B      |
| •   DNF                | 11A           |
| •   Zypper             | 11A           |
| • Build tools          | 11F           |
| •   Commands           | 11F           |
| • make                 | 11F           |
| • make install         | 11F           |
| • ldd                  | 11F           |
| •   Compilers          | 11F           |
| •   Shared libraries   | 11F           |
| • Repositories         | 11D           |
| •   Configuration      | 11D           |
| •   Creation           | 11D           |
| •   Syncing            | 11D           |
| •   Locations          | 11D           |
| • Acquisition commands | 11E           |
| • wget                 | 11E           |
| • curl                 | 11E           |

### 2.2 Given a scenario, manage users and groups.

|                |        |
|----------------|--------|
| • Creation     | 2B, 2C |
| •   useradd    | 2B     |
| •   groupadd   | 2C     |
| • Modification | 2B, 2C |

| Domain and Objective                                             | Covered In |
|------------------------------------------------------------------|------------|
| • usermod                                                        | 2B         |
| • groupmod                                                       | 2C         |
| • passwd                                                         | 2B         |
| • chage                                                          | 2B         |
| • Deletion                                                       | 2B, 2C     |
| • userdel                                                        | 2B         |
| • groupdel                                                       | 2C         |
| • Queries                                                        | 2D         |
| • id                                                             | 2D         |
| • whoami                                                         | 2D         |
| • who                                                            | 2D         |
| • w                                                              | 2D         |
| • last                                                           | 2D         |
| • Quotas                                                         | 4F         |
| • User quota                                                     | 4F         |
| • Group quota                                                    | 4F         |
| • Profiles                                                       | 2E         |
| • Bash parameters                                                | 2E         |
| • User entries                                                   | 2E         |
| • .bashrc                                                        | 2E         |
| • .bash_profile                                                  | 2E         |
| • .profile                                                       | 2E         |
| • Global entries                                                 | 2E         |
| • /etc/bashrc                                                    | 2E         |
| • /etc/profile.d                                                 | 2E         |
| • /etc/skel                                                      | 2E         |
| • /etc/profile                                                   | 2E         |
| • Important files and file contents                              | 2B, 2C     |
| • /etc/passwd                                                    | 2B         |
| • /etc/group                                                     | 2C         |
| • /etc/shadow                                                    | 2B         |
| <b>2.3 Given a scenario, create, modify, and redirect files.</b> |            |
| • Text editors                                                   | 5A         |
| • nano                                                           | 5A         |
| • vi                                                             | 5A         |
| • File readers                                                   | 5C, 5D     |
| • grep                                                           | 5D         |
| • cat                                                            | 5C         |
| • tail                                                           | 5C         |
| • head                                                           | 5C         |
| • less                                                           | 5C         |
| • more                                                           | 5C         |
| • Output redirection                                             | 5E         |
| • <                                                              | 5E         |

| Domain and Objective            | Covered In          |
|---------------------------------|---------------------|
| • >                             | 5E                  |
| •                               | 5E                  |
| • <<                            | 5E                  |
| • >>                            | 5E                  |
| • 2>                            | 5E                  |
| • &>                            | 5E                  |
| • stdin                         | 5E                  |
| • stdout                        | 5E                  |
| • stderr                        | 5E                  |
| • /dev/null                     | 5E                  |
| • /dev/tty                      | 5E                  |
| • xargs                         | 5E                  |
| • tee                           | 5E                  |
| • Here documents                | 5E                  |
| • Text processing               | 5D, 5E              |
| • grep                          | 5D, 5E              |
| • tr                            | 5D                  |
| • echo                          | 5D                  |
| • sort                          | 5D                  |
| • awk                           | 5D                  |
| • sed                           | 5D                  |
| • cut                           | 5D                  |
| • printf                        | 5D                  |
| • egrep                         | 5D                  |
| • wc                            | 5D                  |
| • paste                         | 5D                  |
| • File and directory operations | 4A, 5B, 5C, 5D, 12F |
| • touch                         | 5C                  |
| • mv                            | 5C                  |
| • cp                            | 5C                  |
| • rm                            | 5C                  |
| • scp                           | 12F                 |
| • ls                            | 5C                  |
| • rsync                         | 12F                 |
| • mkdir                         | 5C                  |
| • rmdir                         | 5C                  |
| • ln                            | 5D                  |
| • Symbolic (soft)               | 5D                  |
| • Hard                          | 5D                  |
| • unlink                        | 5C                  |
| • inodes                        | 4A                  |
| • find                          | 5B                  |
| • locate                        | 5B                  |
| • grep                          | 5D                  |

| Domain and Objective                           | Covered In |
|------------------------------------------------|------------|
| • which                                        | 5B         |
| • whereis                                      | 5B         |
| • diff                                         | 5D         |
| • updatedb                                     | 5B         |
| <b>2.4 Given a scenario, manage services.</b>  |            |
| • systemd management                           | 8C, 8D     |
| • systemctl                                    | 8C         |
| • enabled                                      | 8C         |
| • disabled                                     | 8C         |
| • start                                        | 8C         |
| • stop                                         | 8C         |
| • mask                                         | 8C         |
| • restart                                      | 8C         |
| • status                                       | 8C         |
| • daemon-reload                                | 8C         |
| • systemd-analyze blame                        | 8D         |
| • Unit files                                   | 8C         |
| • Directory locations                          | 8C         |
| • Environment parameters                       | 8C         |
| • Targets                                      | 8C         |
| • hostnamectl                                  | 8C         |
| • automount                                    | 8C         |
| • SysVinit                                     | 8C         |
| • chkconfig                                    | 8C         |
| • on                                           | 8C         |
| • off                                          | 8C         |
| • level                                        | 8C         |
| • Runlevels                                    | 8C         |
| • Definitions of 0–6                           | 8C         |
| • /etc/init.d                                  | 8C         |
| • /etc/rc.d                                    | 8C         |
| • /etc/rc.local                                | 8C         |
| • /etc/inittab                                 | 8C         |
| • Commands                                     | 8C         |
| • runlevel                                     | 8C         |
| • telinit                                      | 8C         |
| • service                                      | 8C         |
| • restart                                      | 8C         |
| • status                                       | 8C         |
| • stop                                         | 8C         |
| • start                                        | 8C         |
| • reload                                       | 8C         |
| <b>2.5 Summarize and explain server roles.</b> |            |
| • NTP                                          | 10B        |

| <b>Domain and Objective</b>                                | <b>Covered In</b> |
|------------------------------------------------------------|-------------------|
| • SSH                                                      | 10B               |
| • Web                                                      | 10B               |
| • Certificate authority                                    | 10B               |
| • Name server                                              | 10B               |
| • DHCP                                                     | 10B               |
| • File servers                                             | 10B               |
| • Authentication server                                    | 10B               |
| • Proxy                                                    | 10B               |
| • Logging                                                  | 10B               |
| • Containers                                               | 10B               |
| • VPN                                                      | 10B               |
| • Monitoring                                               | 10B               |
| • Database                                                 | 10B               |
| • Print server                                             | 10B               |
| • Mail server                                              | 10B               |
| • Load balancer                                            | 10B               |
| • Clustering                                               | 10B               |
| <b>2.6 Given a scenario, automate and schedule jobs.</b>   |                   |
| • cron                                                     | 14A               |
| • at                                                       | 14A               |
| • crontab                                                  | 14A               |
| • fg                                                       | 8D                |
| • bg                                                       | 8D                |
| • &                                                        | 8D                |
| • kill                                                     | 8D                |
| • Ctrl+C                                                   | 8D                |
| • Ctrl+Z                                                   | 8D                |
| • nohup                                                    | 8D                |
| <b>2.7 Explain the use and operation of Linux devices.</b> |                   |
| • Types of devices                                         | 9A                |
| • Client devices                                           | 9A                |
| • Bluetooth                                                | 9A                |
| • Wi-Fi                                                    | 9A                |
| • USB                                                      | 9A                |
| • Monitors                                                 | 9A                |
| • GPIO                                                     | 9A                |
| • Network adapters                                         | 9A                |
| • PCI                                                      | 9A                |
| • HBA                                                      | 9A                |
| • SATA                                                     | 9A                |
| • SCSI                                                     | 9A                |
| • Printers                                                 | 9A                |
| • Video                                                    | 9A                |
| • Audio                                                    | 9A                |



| <b>Domain and Objective</b>                                                                                | <b>Covered In</b>      |
|------------------------------------------------------------------------------------------------------------|------------------------|
| • SSH port forwarding                                                                                      | 8B                     |
| • Local                                                                                                    | 8B                     |
| • Remote                                                                                                   | 8B                     |
| • X11 forwarding                                                                                           | 8B                     |
| • VNC                                                                                                      | 8B                     |
| • Accessibility                                                                                            | 8B                     |
| <b>Domain 3.0 Security</b>                                                                                 |                        |
| <b>3.1 Given a scenario, apply or acquire the appropriate user and/or group permissions and ownership.</b> |                        |
| • File and directory permissions                                                                           | 2B, 3A, 3B, 3C, 4F, 5C |
| • Read, write, execute                                                                                     | 3A                     |
| • User, group, other                                                                                       | 3A                     |
| • SUID 3C                                                                                                  |                        |
| • Octal notation                                                                                           | 3A                     |
| • umask                                                                                                    | 3A                     |
| • Sticky bit                                                                                               | 3C                     |
| • SGID 3C                                                                                                  |                        |
| • Inheritance                                                                                              | 3C                     |
| • Utilities                                                                                                | 2B, 3A, 3B, 3C, 4F, 5C |
| • chmod                                                                                                    | 3A                     |
| • chown                                                                                                    | 3B                     |
| • chgrp                                                                                                    | 3B                     |
| • getfacl                                                                                                  | 3C                     |
| • setfacl                                                                                                  | 3C                     |
| • ls                                                                                                       | 5C                     |
| • ulimit                                                                                                   | 4F                     |
| • chage                                                                                                    | 2B                     |
| • Context-based permissions                                                                                | 12C                    |
| • SELinux configurations                                                                                   | 12C                    |
| • disabled                                                                                                 | 12C                    |
| • permissive                                                                                               | 12C                    |
| • enforcing                                                                                                | 12C                    |
| • SELinux policy                                                                                           | 12C                    |
| • targeted                                                                                                 | 12C                    |
| • SELinux tools                                                                                            | 12C                    |
| • setenforce                                                                                               | 12C                    |
| • getenforce                                                                                               | 12C                    |
| • sestatus                                                                                                 | 12C                    |
| • setsebool                                                                                                | 12C                    |
| • getsebool                                                                                                | 12C                    |
| • chcon                                                                                                    | 12C                    |
| • restorecon                                                                                               | 12C                    |
| • ls -Z                                                                                                    | 12C                    |
| • ps -Z                                                                                                    | 12C                    |

| Domain and Objective                                                                                | Covered In |
|-----------------------------------------------------------------------------------------------------|------------|
| • AppArmor                                                                                          | 12C        |
| • aa-disable                                                                                        | 12C        |
| • aa-complain                                                                                       | 12C        |
| • aa-unconfined                                                                                     | 12C        |
| • /etc/apparmor.d/                                                                                  | 12C        |
| • /etc/apparmor.d/tunables                                                                          | 12C        |
| • Privilege escalation                                                                              | 2A         |
| • su                                                                                                | 2A         |
| • sudo                                                                                              | 2A         |
| • wheel                                                                                             | 2A         |
| • visudo                                                                                            | 2A         |
| • sudoedit                                                                                          | 2A         |
| • User types                                                                                        | 2A         |
| • Root                                                                                              | 2A         |
| • Standard                                                                                          | 2A         |
| • Service                                                                                           | 2A         |
| <b>3.2 Given a scenario, configure and implement appropriate access and authentication methods.</b> |            |
| • PAM                                                                                               | 12B        |
| • Password policies                                                                                 | 12B        |
| • LDAP integration                                                                                  | 12B        |
| • User lockouts                                                                                     | 12B        |
| • Required, allowed, or sufficient                                                                  | 12B        |
| • /etc/pam.d/                                                                                       | 12B        |
| • pam_tally2                                                                                        | 12B        |
| • faillock                                                                                          | 12B        |
| • SSH                                                                                               | 12B        |
| • ~/.ssh/                                                                                           | 12B        |
| • known_hosts                                                                                       | 12B        |
| • authorized_keys                                                                                   | 12B        |
| • config                                                                                            | 12B        |
| • id_rsa                                                                                            | 12B        |
| • id_rsa.pub                                                                                        | 12B        |
| • User-specific access                                                                              | 12B        |
| • TCP wrappers                                                                                      | 12B        |
| • /etc/ssh/                                                                                         | 12B        |
| • ssh_config                                                                                        | 12B        |
| • sshd_config                                                                                       | 12B        |
| • ssh-copy-id                                                                                       | 12B        |
| • ssh-keygen                                                                                        | 12B        |
| • ssh-add                                                                                           | 12B        |
| • TTYs                                                                                              | 12B        |
| • /etc/securetty                                                                                    | 12B        |
| • /dev/tty#                                                                                         | 12B        |

| Domain and Objective                                                                                                                                                                                                                                                       | Covered In                                                                 |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| • PTYs                                                                                                                                                                                                                                                                     | 12B                                                                        |
| • PKI <ul style="list-style-type: none"> <li>• Self-signed</li> <li>• Private keys</li> <li>• Public keys</li> <li>• Hashing</li> <li>• Digital signatures</li> <li>• Message digest</li> </ul>                                                                            | 12A, 12B                                                                   |
| • VPN as a client <ul style="list-style-type: none"> <li>• SSL/TLS</li> <li>• Transport mode</li> <li>• Tunnel mode</li> <li>• IPSec</li> <li>• DTLS</li> </ul>                                                                                                            | 12B                                                                        |
| • Boot security <ul style="list-style-type: none"> <li>• Boot loader password</li> <li>• UEFI/BIOS password</li> </ul>                                                                                                                                                     | 12A                                                                        |
| • Additional authentication methods <ul style="list-style-type: none"> <li>• Multi-factor authentication               <ul style="list-style-type: none"> <li>• Tokens</li> <li>• Hardware</li> <li>• Software</li> </ul> </li> <li>• OTP</li> <li>• Biometrics</li> </ul> | 12A                                                                        |
| • RADIUS                                                                                                                                                                                                                                                                   | 12A                                                                        |
| • TACACS+                                                                                                                                                                                                                                                                  | 12A                                                                        |
| • LDAP                                                                                                                                                                                                                                                                     | 12A                                                                        |
| • Kerberos                                                                                                                                                                                                                                                                 | 12A                                                                        |
|                                                                                                                                                                                                                                                                            | <ul style="list-style-type: none"> <li>• kinit</li> <li>• klist</li> </ul> |
| • Importance of disabling root login via SSH                                                                                                                                                                                                                               | 12A                                                                        |
| • Password-less login <ul style="list-style-type: none"> <li>• Enforce use of PKI</li> </ul>                                                                                                                                                                               | 12A                                                                        |
| • chroot jail services                                                                                                                                                                                                                                                     | 12A                                                                        |
| • No shared IDs                                                                                                                                                                                                                                                            | 12A                                                                        |
| • Importance of denying hosts                                                                                                                                                                                                                                              | 12A, 12B                                                                   |
| • Separation of OS data from application data <ul style="list-style-type: none"> <li>• Disk partition to maximize system availability</li> </ul>                                                                                                                           | 12A                                                                        |
| • Change default ports                                                                                                                                                                                                                                                     | 12A, 12D                                                                   |
| • Importance of disabling or uninstalling unused and unsecure services <ul style="list-style-type: none"> <li>• FTP</li> </ul>                                                                                                                                             | 12A                                                                        |

### 3.3 Summarize security best practices in a Linux environment.

- Boot security
  - Boot loader password
  - UEFI/BIOS password
- Additional authentication methods
  - Multi-factor authentication
    - Tokens
    - Hardware
    - Software
  - OTP
  - Biometrics
- RADIUS
- TACACS+
- LDAP
- Kerberos
  - kinit
  - klist
- Importance of disabling root login via SSH
- Password-less login
  - Enforce use of PKI
- chroot jail services
- No shared IDs
- Importance of denying hosts
- Separation of OS data from application data
  - Disk partition to maximize system availability
- Change default ports
- Importance of disabling or uninstalling unused and unsecure services
  - FTP

| Domain and Objective                                                  | Covered In                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| • Telnet                                                              | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Finger                                                              | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Sendmail                                                            | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Postfix                                                             | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Importance of enabling SSL/TLS                                      | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Importance of enabling auditd                                       | 12A <ul style="list-style-type: none"> <li>• CVE monitoring</li> </ul>                                                                                                                                                                                                                                                                   |
| • Discouraging use of USB devices                                     | 12A <ul style="list-style-type: none"> <li>• Disk encryption</li> </ul>                                                                                                                                                                                                                                                                  |
| • LUKS                                                                | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Restrict cron access                                                | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Disable Ctrl+Alt+Del                                                | 12A                                                                                                                                                                                                                                                                                                                                      |
| • Add banner                                                          | 12A <ul style="list-style-type: none"> <li>• MOTD</li> </ul>                                                                                                                                                                                                                                                                             |
|                                                                       | 12A                                                                                                                                                                                                                                                                                                                                      |
| <b>3.4 Given a scenario, implement logging services.</b>              |                                                                                                                                                                                                                                                                                                                                          |
| • Key file locations                                                  | 12E <ul style="list-style-type: none"> <li>• /var/log/secure</li> <li>• /var/log/messages</li> <li>• /var/log/[application]</li> <li>• /var/log/kern.log</li> <li>• Log management</li> <li>• Third-party agents</li> <li>• logrotate</li> <li>• /etc/rsyslog.conf</li> <li>• journald</li> <li>• journalctl</li> <li>• lastb</li> </ul> |
|                                                                       | 12E                                                                                                                                                                                                                                                                                                                                      |
| <b>3.5 Given a scenario, implement and configure Linux firewalls.</b> |                                                                                                                                                                                                                                                                                                                                          |
| • Access control lists                                                | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Source                                                              | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Destination                                                         | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Ports                                                               | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Protocol                                                            | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Logging                                                             | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Stateful vs. stateless                                              | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Accept                                                              | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Reject                                                              | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Drop                                                                | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Log                                                                 | 12D <ul style="list-style-type: none"> <li>• Technologies</li> </ul>                                                                                                                                                                                                                                                                     |
| • firewalld                                                           | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Zones                                                               | 12D                                                                                                                                                                                                                                                                                                                                      |
| • Run time                                                            | 12D                                                                                                                                                                                                                                                                                                                                      |

| Domain and Objective                         | Covered In |
|----------------------------------------------|------------|
| • iptables                                   | 12D        |
| • Persistency                                | 12D        |
| • Chains                                     | 12D        |
| • ufw                                        | 12D        |
| • /etc/default/ufw                           | 12D        |
| • /etc/ufw/                                  | 12D        |
| • Netfilter                                  | 12D        |
| • IP forwarding                              | 12D        |
| • /proc/sys/net/ipv4/ip_forward              | 12D        |
| • /proc/sys/net/ipv6/conf/all/forwarding     | 12D        |
| • Dynamic rule sets                          | 12D        |
| • DenyHosts                                  | 12D        |
| • Fail2ban                                   | 12D        |
| • IPset                                      | 12D        |
| • Common application firewall configurations | 12D        |
| • /etc/services                              | 12D        |
| • Privileged ports                           | 12D        |

### 3.6 Given a scenario, back up, restore, and compress files.

- Archive and restore utilities
  - tar 12F
  - cpio 12F
  - dd 12F
- Compression
  - gzip 12F
  - xz 12F
  - bzip2 12F
  - zip 12F
- Backup types
  - Incremental 12F
  - Full 12F
- Snapshot clones 12F
- Differential 12F
- Image 12F
- Off-site/off-system storage 12F
  - SFTP 12F
  - SCP 12F
  - rsync 12F
- Integrity checks
  - MD5 12F
  - SHA 12F

### Domain 4.0 Linux Troubleshooting and Diagnostics

#### 4.1 Given a scenario, analyze system properties and remediate accordingly.



| Domain and Objective                  | Covered In |
|---------------------------------------|------------|
| • LVM tools                           | 4B         |
| • partprobe                           | 4A         |
| • CPU monitoring and configuration    | 8E         |
| • /proc/cpuinfo                       | 8E         |
| • uptime                              | 8E         |
| • load average                        | 8E         |
| • sysctl                              | 6B, 8E     |
| • Memory monitoring and configuration | 8E         |
| • swapon                              | 8E         |
| • swapoff                             | 8E         |
| • mkswap                              | 8E         |
| • vmstat                              | 8E         |
| • Out of memory killer                | 8E         |
| • /proc/meminfo                       | 8E         |
| • Buffer cache output                 | 8E         |
| • Lost root password                  | 8C         |
| • Single user mode                    | 8C         |

#### 4.2 Given a scenario, analyze system processes in order to optimize performance.

|                         |         |
|-------------------------|---------|
| • Process management    | 8D, 13A |
| • Process states        | 8D      |
| • Zombie                | 8D      |
| • Uninterruptible sleep | 8D      |
| • Interruptible sleep   | 8D      |
| • Running               | 8D      |
| • Priorities            | 8D      |
| • Kill signals          | 8D      |
| • Commands              | 8D, 13A |
| • nice                  | 8D      |
| • renice                | 8D      |
| • top                   | 8D      |
| • time                  | 13A     |
| • ps                    | 8D      |
| • lsof                  | 8D      |
| • pgrep                 | 8D      |
| • pkill                 | 8D      |
| • PIDs                  | 8D      |

#### 4.3 Given a scenario, analyze and troubleshoot user issues.

|               |    |
|---------------|----|
| • Permissions | 3D |
| • File        | 3D |

| Domain and Objective                                                                   | Covered In |
|----------------------------------------------------------------------------------------|------------|
| • Directory                                                                            | 3D         |
| • Access                                                                               | 12B        |
| • Local                                                                                | 12B        |
| • Remote                                                                               | 12B        |
| • Authentication                                                                       | 12B, 12C   |
| • Local                                                                                | 12B        |
| • External                                                                             | 12B        |
| • Policy violations                                                                    | 12B, 12C   |
| • File creation                                                                        | 3D, 4F     |
| • Quotas                                                                               | 4F         |
| • Storage                                                                              | 4F         |
| • Inode exhaustion                                                                     | 4F         |
| • Immutable files                                                                      | 3D         |
| • Insufficient privileges for authorization                                            | 12C        |
| • SELinux violations                                                                   | 12C        |
| • Environment and shell issues                                                         | 13A        |
| <b>4.4 Given a scenario, analyze and troubleshoot application and hardware issues.</b> |            |
| • SELinux context violations                                                           | 12C        |
| • Storage                                                                              | 4F, 9D     |
| • Degraded storage                                                                     | 4F         |
| • Missing devices                                                                      | 4F         |
| • Missing volumes                                                                      | 4F         |
| • Missing mount point                                                                  | 4F         |
| • Performance issues                                                                   | 4F         |
| • Resource exhaustion                                                                  | 4F         |
| • Adapters                                                                             | 9D         |
| • SCSI                                                                                 | 9D         |
| • RAID                                                                                 | 9D         |
| • SATA                                                                                 | 9D         |
| • HBA                                                                                  | 9D         |
| • /sys/class/scsi_host/host#/scan                                                      | 9D         |
| • Storage integrity                                                                    | 4F         |
| • Bad blocks                                                                           | 4F         |
| • Firewall                                                                             | 12D        |
| • Restrictive ACLs                                                                     | 12D        |
| • Blocked ports                                                                        | 12D        |
| • Blocked protocols                                                                    | 12D        |
| • Permission                                                                           | 2A, 3C, 3D |
| • Ownership                                                                            | 3D         |
| • Executables                                                                          | 3D         |
| • Inheritance                                                                          | 3C, 3D     |
| • Service accounts                                                                     | 2A, 3D     |
| • Group memberships                                                                    | 3D         |

| Domain and Objective                         | Covered In |
|----------------------------------------------|------------|
| • Dependencies                               | 11G        |
| • Patching                                   | 11G        |
| • Update issues                              | 11G        |
| • Versioning                                 | 11G        |
| • Libraries                                  | 11G        |
| • Environment variables                      | 13A        |
| • GCC compatibility                          | 11G        |
| • Repositories                               | 11G        |
| • Troubleshooting additional hardware issues | 9D         |
| • Memory                                     | 9D         |
| • Printers                                   | 9D         |
| • Video                                      | 9D         |
| • GPU drivers                                | 9D         |
| • Communications ports                       | 9D         |
| • USB                                        | 9D         |
| • Keyboard mapping                           | 9D         |
| • Hardware or software compatibility issues  | 9D         |
| • Commands                                   | 9D         |
| • dmidecode                                  | 9D         |
| • lshw                                       | 9D         |

## 5.0 Automation and Scripting

### 5.1 Given a scenario, deploy and execute basic Bash scripts.

- Shell environments and shell variables
  - PATH 13A
  - Global 13A
  - Local 13A
  - export 13A
  - env 13A
  - set 13A, 13B
  - printenv 13A
  - echo 13A
- #!/bin/bash 13C
- Sourcing scripts 13C
- Directory and file permissions 13C
  - chmod 13C
- Extensions 13C
- Commenting 13B
  - # 13B
- File globbing 13C
- Shell expansions 13C
  - \${} 13C
  - \${0} 13C
  - `` 13C

| Domain and Objective                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Covered In                                                                              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Redirection and piping</li> <li>• Exit codes</li> <li>• stderr</li> <li>• stdin</li> <li>• stdout           <ul style="list-style-type: none"> <li>• Metacharacters</li> <li>• Positional parameters</li> <li>• Looping constructs</li> </ul> </li> <li>• while</li> <li>• for</li> <li>• until           <ul style="list-style-type: none"> <li>• Conditional statements</li> </ul> </li> <li>• if</li> <li>• case           <ul style="list-style-type: none"> <li>• Escaping characters</li> </ul> </li> </ul> | 13C<br>13C<br>13C<br>13C<br>13C<br>13D<br>13D<br>13D<br>13D<br>13D<br>13D<br>13D<br>13B |
| <b>5.2 Given a scenario, carry out version control using Git.</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                         |
| <ul style="list-style-type: none"> <li>• Arguments</li> <li>• clone</li> <li>• push</li> <li>• pull</li> <li>• commit</li> <li>• merge</li> <li>• branch</li> <li>• log</li> <li>• init</li> <li>• config           <ul style="list-style-type: none"> <li>• Files</li> </ul> </li> <li>• .gitignore</li> <li>• .git/</li> </ul>                                                                                                                                                                                                                           | 14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B<br>14B |
| <b>5.3 Summarize orchestration processes and concepts.</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                         |
| <ul style="list-style-type: none"> <li>• Agent</li> <li>• Agentless</li> <li>• Procedures</li> <li>• Attributes</li> <li>• Infrastructure automation</li> <li>• Infrastructure as code</li> <li>• Inventory</li> <li>• Automated configuration management</li> <li>• Build automation</li> </ul>                                                                                                                                                                                                                                                           | 14C<br>14C<br>14C<br>14C<br>14C<br>14C<br>14C<br>14C<br>14C                             |

# Solutions

---

## Activity 1-1: Discussing the Linux Design Philosophy

---

**1. What is the distinction between free software and open source software given by Richard Stallman, the founder of the free software movement?**

Open source is a development methodology; free software is a social movement. Free software refers to freedom rather than to software that has no cost associated with it.

**2. What is one of the main advantages of Linux that has enabled it to be applied to many different computing roles?**

Linux is highly extensible, which makes it an excellent choice for computing roles as diverse as servers, workstations, mainframes, and mobile devices.

**3. In which four major computing roles does Linux dominate the market?**

Linux dominates in mobile devices where it commands over 80% of the market; in Internet servers such as domain hosting and Amazon's EC2 platform (>90%); in embedded computing; and in mainframes/supercomputers.

**4. True or false? There is a single Linux master distribution from which all other Linux sub-distributions are derived.**

False. There is no single master Linux distribution. There are several top-level (not derived from another distribution) distributions from which others are derived. Slackware, Debian, and Red Hat Enterprise Linux are examples of top-level distributions.

**5. Which Unix philosophy states that developers should create software that is limited in functionality because it's easier to use and to maintain?**

Worse is better. The assumption is that limited functionality is worse than creating software that does everything.

---

## Activity 1-2: Identifying the Linux Design Philosophy

---

**1. Linux incorporates software tools from which of the following sources?**

- Unix
- The GNU Project
- Berkeley Software Distribution (BSD)
- macOS

**2. Which of the following statements about free and open source software (FOSS) are true? (Choose two.)**

- Anyone can access the source code.
- The author cannot charge a price for the source code.
- Anyone can modify the source code.
- The source code can only be distributed to a public developer community.

**3. Which of the following FOSS licenses is the Linux kernel released under?**

- GNU General Public License (GPL) version 2
- GNU GPL version 3
- MIT License
- Apache License 2.0

**4. Which of the following best describes a typical Linux distribution?**

- The Linux kernel with proprietary software tools.
- The Linux kernel with GNU software tools.
- The Linux kernel only.
- GNU software tools only.

**5. True or false? Linux distributions can include proprietary software by default.**

- True
- False

**6. Which of the following Linux distributions is the no-cost version of Red Hat Enterprise Linux (RHEL)?**

- Ubuntu
- Fedora
- openSUSE
- CentOS

**7. Which of the following characteristics are emphasized by the Unix design philosophy? (Choose two.)**

- Modularity
- Stability
- Complexity
- Simplicity

**8. True or false? Linux is Unix-like in its design.**

- True
- False

**9. Which of the following computing roles does Linux dominate in market share? (Choose three.)**

- Workstations
- Web servers
- Mobile devices
- Supercomputers

## Activity 1-3: Discussing Shell Commands

**1. True or false? In Linux, Equipment.txt and equipment.txt refer to the same file.**

False. Linux file names are case-sensitive. These are two different files.

**2. Which command enables a user to log in to a system with their own credentials, and then to become the root user to perform administrative tasks?**

SU —The SU command enables an authorized user to become the root user.

**3. A fellow administrator calls you and needs you to check the /etc/shadow file for a new user's entry, confirming that it indeed exists. How would you look at the contents of the /etc/shadow file without the possibility of changing the file?**

1) SU – 2) cat /etc/shadow —You first have to become the root user because the /etc/shadow file can only be read as root. To look at the file without the possibility of altering its contents, use the **Cat** command to list its contents to the screen.

**4. You need to quickly create an empty file (log.txt) so that an application can copy data into it. Which command can you use to create the file?**

**touch log.txt**

- 5. You recall that a few days ago, you ran a command that helped you find a filtered list of files on the file system. You don't remember the command's somewhat complex syntax. How can you reproduce the command without searching the Internet for the correct syntax again?**

Use the **Up Arrow** key until you find it and then press **Enter** to execute it. The **Up Arrow** key scrolls back, one command at a time, through your command history.

## Activity 1-5: Discussing Help in Linux

- 1. Since Linux is generally community (user) supported, name three sources of Linux documentation.**

Any three of the following: Manual pages, built-in help commands, online documentation projects, Usenet newsgroups, Internet mailing lists, question and answer websites, forums and social media, and books and other print resources.

- 2. To access complete local documentation, which command can you enter at the command-line to access extensive usage information on the cp (copy) command?**

`man cp`

- 3. A coworker sends you an instant message and asks you how to appropriately format a manual page for the mv command so that she can send it to the printer and get readable output. What command can she enter?**

`man -t mv`

- 4. Which online documentation site provides a comprehensive resource for Linux documentation including manual pages, FAQs, HOWTOs, and other types of documentation?**

The Linux Documentation Project (<https://www.tldp.org>)

- 5. If you know the name of a command and want to know what its function is, which command would you use to find out?**

`whatis`

## Activity 1-6: Accessing Help in Linux

- 3. Looking at these results, which command(s) do you think would best fulfill the capabilities that you're looking for?**

Answers may vary, but one of the `grep` variants is likely the most appropriate command. The `awk` command and its variants could be helpful, but appear to be more advanced.

- 5. Given what you've read in the man page for grep so far, answer what you think the following command does:** `grep -i hello myfile`

This command will return any lines in the file `myfile` that contain the text "hello", no matter what case the text is in (case insensitive).

## 7. How confident are you that this command fulfills what you're looking for?

Answers may vary, but the `grep` command does generally meet your requirements. However, this doesn't mean it's the only command, or the best command, for the job.

## 8. You still want to learn more about other commands that your team could use to search the contents of a text file. Aside from the help options built into Linux, what other sources can you consult in your research?

Answers may vary, but there is a wide variety of useful sources on the Internet that could help you find what you're looking for. You could pose specific answers to Q&A sites like Stack Exchange; ask discussion questions on newsgroups, mailing lists, and forums dedicated to Linux support; consult supplementary and/or advanced documentation through a resource like the Linux Documentation Project; or consult distro-specific documentation on the relevant distro's website.

## Activity 2-1: Discussing Superuser Privileges

### 1. Name the three types of Linux user accounts.

Root user (superuser), standard user, and service.

### 2. You need to add an administrator to the /etc/sudoers file to give them the ability to use the sudo command. Which command must you use to add users to the /etc/sudoers file?

You must use `visudo` since you cannot edit the `/etc/sudoers` file directly with Vim or other text editors.

### 3. Why is it a security best practice to log onto a Linux system with a regular user account rather than with the root user account?

The primary reason is to prevent harmful mistakes from happening to the system through errant commands such as `rm` (remove). Users who can use the `SUDO` command should do so on an individual command basis to perform necessary tasks with elevated privileges, while remaining in their user shells for other functions.

### 4. Why is it important to put the principle of least privilege into practice?

System security is greatly enhanced by only granting users the minimum amount of rights and permissions they require to perform a task.

### 5. Describe the difference between using the su command and using the sudo command to perform administrative tasks.

The `SU` command enables an authorized user to become the root user by switching to the root user account in a complete root-owned shell. This is handy for the administrator but very dangerous because of the potential for human error. The `SUDO` command enables an authorized user to issue individual commands as the root (or other user), limiting potential damage to the system.

---

## Activity 2-5: Discussing Creating, Modifying, and Deleting Users

---

1. In which file are user accounts stored?

/etc/passwd

2. What are the three commands used to properly edit the /etc/ passwd file?

useradd, usermod, and userdel

3. In which file are hashed passwords stored?

/etc/shadow

4. Why are /etc/passwd and /etc/shadow different files?

The /etc/passwd file stores user account details such as user names, default shell, home directory, and group names. The /etc/shadow file contains the hashed passwords that are only readable by the root user account, whereas /etc/passwd is readable by everyone. Having a separate file for each, with different permissions, strengthens security.

5. With which command can you change the default user shell to the KornShell for user bsmith?

sudo usermod -s /bin/ksh bsmith

---

## Activity 2-7: Discussing Creating, Modifying, and Deleting Groups

---

1. Why do administrators, classes, and best practices state that you should manage users by managing groups?

Managing groups simplifies user administration. Rather than granting explicit permissions to users on an individual basis, you grant them to groups and add users to those groups.

2. Which file contains the groups and user members of those groups?

/etc/group

3. Which three commands does a system administrator use to properly edit a Linux system's group file?

groupadd, groupmod, and groupdel

4. Which command does a system administrator use to add a user (bsmith) to an existing group (finance)?

sudo usermod -aG finance bsmith

- 5. True or false? The groupmod command is used to change the name of an existing group.**

True.

## Activity 2-9: Discussing Querying Users and Groups

- 1. As a Linux administrator, you might often switch accounts from yours to the root user or to other user accounts to perform tasks. Which command can you issue to find out which user account you're currently using?**

whoami

- 2. You suspect that a hardware failure is imminent and you need to reboot the system to ensure that everything is working properly or to force a failure. Before issuing the reboot command, which command can you use to check to see if other users are logged onto the system?**

who

- 3. A user reports that she was working on an important script when the system rebooted at approximately 6:30 P.M. last night. No warning was given. How can you find out who was logged into the system at that time and who could have rebooted the system without warning?**

The **last** command will inform you as to who was logged into the system. If an administrator rebooted the system, you'll be able to identify him or her from the entries displayed. If the system crashed, **last** will also provide that information. Further investigation of systems logs may be required in the event of a crash.

- 4. As a system administrator, why might you issue the id command?**

The **id** command displays group information at a glance in an easy-to-read format.

- 5. As an administrator, you might need to reboot a system or otherwise perform maintenance. Which command would you issue to not only view logged on users but also their current activity?**

The **w** command displays idle time and the amount of CPU time consumed by user-owned processes. This information will tell you if it's safe to reboot a system or if you should ask users to log off first.

## Activity 2-11: Discussing Account Profiles

- 1. True or false? Using a dot at the beginning of a file name makes the file more secure.**

False. The file is hidden from normal view as a matter of organization, but does nothing for security.

- 2. What is the primary difference between the .bashrc and the .bash\_profile files?**

The **.bash\_profile** file is executed upon first login to the system, and **.bashrc** is executed upon subsequent logins.

- 3. Which directory's contents are copied to the user's home directory upon account creation?**

/etc/skel

- 4. Which file forces system-wide customizations for all users on a system that a user cannot change?**

/etc/profile

- 5. Where should administrators set system-wide variables on a Linux system rather than editing the /etc/profile file directly?**

In scripts within the /etc/profile.d directory.

## Activity 3-2: Discussing File and Directory Permissions

- 1. Multiple users have complained about file access in a shared directory, but you've checked your daily backup reports and there are no corrupt files. Which command can you issue in the directory in question to investigate the problem further?**

**ls -l**—This command displays the permissions of all files in the directory to help you sort out the problem. It's likely that the directory's group permissions are incorrectly set.

- 2. A group of system administrators were discussing file permissions and decided that setting a particular root-owned text file to read-only for everyone is a best practice. What do the permissions for this file look like?**

Either 664 or rW-rW-r-- is correct.

- 3. A user cannot execute a script (collect.sh) she created and has sent you the contents of the script via email to inspect. After looking at the script, you determine the script is correctly written but permissions are the problem. What command can you issue to adjust the file's permissions as necessary?**

chmod u+x collect.sh

- 4. A user changed the permissions of a script (myscript.sh) in a shared directory. The user is curious why everyone can execute the script if the user owns the script and everyone else only has read access. To make the script executable, what command did the user mistakenly issue?**

**chmod +x myscript.sh** —The permissions changed to rwxrwxr-x or 775, which gives everyone execute permission. To limit execute permission to the user and group only, the command should have been: **chmod ug+x myscript.sh**

- 5. Your team lead is tired of receiving help desk tickets to restore deleted files from a directory that contains hundreds of files and subdirectories. She decides to have you fix the problem by making all of the files read-only. How do you change all the files to read-only without having to traverse each directory?**

`chmod -R 644 *` —This command changes all files in the current directory to 644 recursively.

## Activity 3-4: Discussing File and Directory Ownership

- 1. A user (Bob Smith—username: bsmith) calls you to request that you restore a group of files he accidentally deleted from his home directory. You copy the files for him but he later complains that he can no longer edit the files. What do you need to do so that he can edit his files?**

You need to change ownership of the files to him. Change user and group ownership recursively so that Bob owns all files and directories. For example: `Sudo chown -R bsmith:bsmith *`

- 2. Gina wants to share some marketing files with two other members of her team but doesn't want them to access those files in her home directory. She also wants the directory and its files to only be available to the Marketing group. What steps can you take as an administrator to accomplish this request?**

1) You need to create a new group (mkt): `sudo groupadd mkt` 2) Add users to the group: `sudo usermod -aG mkt gina linda mark` 3) Create the shared directory: `sudo mkdir /opt/marketing` 4) Change group ownership to mkt: `sudo chgrp mkt /opt/marketing` 5) Change permissions so that the Marketing group has full control of the directory and its contents and remove everyone else: `sudo chmod 770 /opt/marketing`

- 3. What command is equivalent to issuing `chown :mygrp file1`?**

`chgrp mygrp file1` —Both commands change the file's group to `mygrp` without changing the owner.

- 4. As an administrator, you've created a shared directory and made the necessary permissions changes to allow a group of users access to that directory. You've also added users to the group. When a user (susan) who is a member of the group creates a file in the shared directory, what user and group permissions will the new file have?**

The file's user and group ownership will both be `SUSAN`.

- 5. What must a user do in a shared directory to ensure that each group member has full read and write access to files they create?**

Change the group ownership to the group: `chgrp accounting salaries.txt` — The user doesn't have to be root or use `SUDO` to change group ownership because the file creator is the file's user and group owner, and therefore may change its permissions at will.

## Activity 3-6: Discussing Special Permissions and Attributes

- The Marketing manager contacts you stating that the shared directory you set up for the Marketing group works, but not exactly like they'd planned. When one of the group members creates a file in the directory, the file takes on the user's user and group permissions. For example, when Linda creates a new file, the permissions are -rw- rw-r-- linda linda. The manager wants the files to all retain the mkt group permission, if possible, rather than having the users change the group themselves. What action can you take to fulfill this request?**

`sudo chmod -R g+s /opt/marketing` —By setting the SGID on the directory, every file created by anyone in the `mkt` group will have the `mkt` group ownership.

- Gina, a member of the Marketing group, has decided that she wants her files protected so that only she can delete them, although other Marketing group members need to be able to work on and edit the files. What can she do to fix the problem of others deleting her files?**

You can show Gina the following command to set the sticky bit on her files:

`chmod +t filename.txt` —This command also works on directories.

- Ruth has searched for a solution to her problem: A few of her training documents keep getting changed or removed by system administrators removing files that haven't been accessed in excess of 180 days. She has found that a file can be made immutable, but she cannot make her own files immutable and needs your assistance. How can you make the files `/home/ruth/training1_doc.txt` and `/home/ruth/training2_doc.txt` immutable?**

`sudo chattr +i /home/ruth/training*_doc.txt`

- A user, John, opened a ticket complaining that he has files in his home directory that he cannot remove, although they are his files and he is the user and group owner. He requests that you remove the files `/home/john/billing1.txt`, `billing2.txt`, and `summary.txt`. However, when you attempt to remove the files, you receive an error that you cannot remove the files as the root user. What is a possible resolution for John?**

You can issue the `lsattr` command to see if immutable flag has been set on those files. If it has, you can resolve the problem by removing the immutable flag and then removing the files: 1) `sudo chattr -i /home/john/billing1.txt` 2) `sudo chattr -i /home/john/billing2.txt` 3) `sudo chattr -i /home/john/summary.txt` 4) `sudo rm /home/john/billing1.txt` 5) `sudo rm /home/john/billing2.txt` 6) `sudo rm /home/john/summary.txt`

- 5. You created a shared directory for the Marketing planners, Linda and Mark, named /opt/MPlans. Their group, mplan, has exclusive access to this directory. No other user can access the directory or its contents. Linda decides that Gina needs read-only access to a single file, history.txt, inside the /opt/MPlans directory. Is this kind of restrictive access possible? If so, how can you grant it to Gina?**

Yes, it is possible through ACLs. First, grant Gina read and execute access to the directory: `setfacl -m u:gina:rx /opt/MPlans` and then, set read access to the `history.txt` file inside the `MPlans` directory: `setfacl -m u:gina:r /opt/MPlans/history.txt` —You can check your work by executing `getfacl` on the directory and its contents.

## Activity 3-10: Troubleshooting Permissions Issues

- 1. Rose Stanley opened a ticket indicating that she is denied the ability to delete a directory. She confirmed with the service desk that she has write permission to the directory. What suggestion would you make to address the issue?**

Users also need the execute permission to remove a directory.

- 2. Jerry Robinson received an access denied message on a file. He told the service desk that the user permissions indicate the owner has read access. What suggestion would you make to address the issue?**

Check to see if Jerry Robinson's account is the owner of the file, and if not, make him the owner if he should be.

- 3. Rose Stanley received an access denied message on a file. She used the `ls -l` command to discover what group had been granted access to the file. She believed she should be able to access the file. What suggestion would you make to address the issue?**

Use the `id` command with Rose Stanley's account to ensure she is a member of the group specified by the file's permissions.

- 4. Jerry Robinson cannot use the `cd` command to navigate into a particular directory. He told the service desk that he used `ls -l` to see what group is associated with the file. He also confirmed that the group is listed as having read access to the directory. He then used the `id` command to confirm he is a member of that group. What suggestion would you make to address this issue?**

Users also need the execute permission to change into a directory.

- 5. A group with a shared directory set up by another administrator is having several problems with permissions in that directory. You've been asked to investigate. Which commands can you use to check existing permissions to help assess the current setup?**

**1) `ls -al` 2) `getfacl` 3) `lsattr`** —These commands will give you a good perspective on permissions, ACLs, and any attributes that have been set on the shared directory and files.

- 6. A user complains that a script that she created and has full access to (`rwxrwx---`) is not executing for her. What is the problem with this script?**

The execute bit was removed from the script by the ACL. It's possible that the script consumed too many resources and an administrator likely removed execute access to it.



- 7. Bob opens a ticket stating that he receives a permission denied error when trying to cd into /opt/Finance. Bob is a member of the FinanceDept group. What is the problem with Bob's access?**

While Bob is part of the FinanceDept group and can access all other FinanceDept group assets, he cannot use /opt/Finance because he doesn't have access via ACLs to that directory.

- 8. The application development team created a new web app and requested that you set up a web server-accessible directory and a service account for the application. After setting up everything for the team, they opened a ticket claiming that the application cannot write logs to the log directory. What is the issue, and how can you correct it?**

As the root user, directories that you create are owned by root. You have to explicitly edit permissions to those directories. Change user and group ownership to the service account for any directory that the service needs to write to. The service account should have read and execute access to all other directories that it accesses. After creating a directory and subdirectories and copying files, it's generally a good practice to issue the following command: `Chown -R webapp:webapp /www/webapp` —You can make permissions adjustments on individual directories and files as needed.

- 9. The Graphics department requests that everyone in the company have access to company logo art to insert into emails, letterheads, and other documents. They do not want anyone outside of their group, however, to have any other access. A few days after requesting the change, other users still cannot access the files. You check permissions on the shared directory to find that the permissions are as follows:**

`drwxrwx--- graphics GraphicsDept 4096 Dec 1  
09:42 logos` —**What command can you issue to fix the problem?**

`sudo chmod o+r logos`

## Activity 4-1: Discussing Partitions

- 1. You're a new system administrator and your manager comes to your cubicle to give you a pop quiz over a few concepts. The first one she asks you is to show her how you would set up a new partition on the second storage drive on a Linux system. Which command can you enter to perform this operation?**

`sudo fdisk /dev/sdb`

- 2. A junior administrator asks for your help with setting up a new Linux file system, /dev/sdb2. He retraces his steps with fdisk, creating the partition and saving the setup. But now, he can't figure out why he can't use the file system—how can you help him?**

After inspecting his work in `fdisk`, you tell him that he must build the file system with the `mkfs` command. You assist him by entering the following command:

`sudo mkfs.ext4 /dev/sdb2`

- 3. You created a new partition, /dev/sdb1, created the file system (XFS), and mounted the file system as /Files. You also created a few directories on /Files for various groups within your organization as shared file spaces. After a maintenance reboot a few days later, users are complaining that their directories and files that you set up no longer exist. What is the problem?**

You didn't make an entry for the new partition in `/etc/fstab`, therefore, the file system didn't mount on boot.

- 4. One of the developers has told your manager that he needs the file system / code rebuilt using the XFS file system type. Which utility can you use?**

`mkfs.xfs`

- 5. You and the other system administrators have a maintenance partition, /maint, that you only want mounted when required for administrative activities. How can you use the file system only when you need it?**

Don't make an entry for `/maint` in the `/etc/fstab` file. Only mount the file system when you need it and then unmount it when finished.

## Activity 4-3: Discussing Logical Volumes

- 1. You need to create a new volume for Human Resources. That group has purchased a new 1 TB drive for their volume. What is the first step in the process of creating the logical volume?**

Create a partition using `fdisk` or a similar tool.

- 2. How can you create a physical volume on /dev/sdd1? `pvcreate /dev/sdd1`**

- 3. How can you remove the physical volume**

`/dev/sde1? pvremove /dev/sde1`

- 4. Which command can you use to create a new volume group, shared, with /dev/sdb1 and /dev/sdd1? And then how do you view your volume groups?**

1) `vgcreate shared /dev/sdb1 /dev/sdd1` 2) `vgdisplay`

- 5. Which command can you use to create a new 400 GB logical volume, finance, from volume group vg\_finance?**

`lvcreate --name finance --size 400G vg_finance`

## Activity 4-5: Discussing Mounting File Systems

- 1. What are the three requirements for mounting a file system?**

1) A device (`/dev/sdb1`, for example) 2) A directory (mount point) 3) Root/superuser privileges.

- 2. What final task do you need to complete in mounting a new file system to have that new file system mount at boot time?**

Add its entry to the /etc/fstab file.

- 3. What is the generic format of the /etc/fstab file?**

<Device or UUID> <Mount point> <File system type> <Mount option> <Dump option> <fsck option>

- 4. How can you unmount the /finance file system?**

sudo umount /finance

- 5. If you issue the following command: sudo umount /share1 and you receive the error that the resource is busy, how do you fix the problem and unmount the file system?**

A program or user, possibly you, is using the /share1 directory. You must close the program or otherwise ensure that it, yourself, or other users are not accessing the file system.

---

## Activity 4-7: Discussing Managing File Systems

---

- 1. What are the steps required to grow the file system**

/dev/data/ datastore1 **from its current 200 GB size to 400 GB?**

1) sudo umount /dev/data/datastore1  
2) sudo e2fsck -f /dev/data/datastore1  
3) sudo lvextend -L400G /dev/data/datastore1  
4) sudo resizefs /dev/data/datastore1  
5) sudo mount /dev/data/datastore1 /data1

- 2. Why should you have to unmount a file system prior to performing an fsck on it?**

You should unmount the file system to prevent damage such as file corruption.

- 3. You know that for ext4 file systems, you use the resizefs utility to grow the file system, but which utility can you use to perform this same action on XFS file systems?**

xfs\_growfs

- 4. Another system administrator appears in your cubicle and asks you to display storage devices and their file systems on server05. Which command can you use to show him the info he needs?**

sudo lsblk -f

- 5. You have a coworker who attempted to grow a file system on a Linux server. His problem is that the file system didn't extend despite entering the following command: sudo lvextend -L150G /dev/shared/graphics —He remounted the file system onto /Graphics but the new space doesn't show up. What is the issue?**

He forgot to issue the **resizefs** command to actually resize the file system. The **lvextend** command extends the partition size but does not affect the file system.

## Activity 4-8: Managing File Systems

- 2. Examine the tree structure. What can you identify about your storage devices?**

Answers may vary depending on the underlying hardware, but generally, the system reserves **sda1** and **sda2** as boot file systems. The **sda3** device is typically a logical volume group created by the system to house the root file system, the home file system, and swap space. Below that should be the two physical volumes that you created, and under each should be one or more logical volumes. Some of the logical volumes might extend across both physical volumes.

## Activity 4-9: Discussing the Linux Directory Structure

- 1. If you cd to the root directory of the file system, name three directories that you'll see there.**

The root of the file system is **/** and some possibilities are: **/etc**, **/home**, **/opt**, **/mnt**, **/media**, **/boot**, **/dev**, **/proc**, **/tmp**, **/var**, **/lib**, **/bin**, **/sbin**, **/sys**, **/root**, and **/usr**

- 2. What standard directory contains system logs?**

**/var/** or, more specifically, **/var/log/**

- 3. How can you list files in a hidden directory?**

Similar to any other directory, either you **cd** into the hidden directory and issue the **ls** command or you explicitly name the hidden directory in your command, such as **ls .ssh**

- 4. Explain the difference between /root and /**

The **/root** directory is the root user's home directory and **/** is the root of the file system.

- 5. If you cd into /opt/log/first/test/test1 and then type cd ~ and press Enter, which directory is now your current working directory?**

Your home directory. If your user name is **bsmith**, this is **/home/bsmith**. The **cd ~** command returns you to your home directory regardless of where you are on the file system.

---

## Activity 4-10: Navigating the Linux Directory Structure

---

**3. Why was no file or directory found, when you were just in a directory with this name?**

This is because you referenced a relative path, which means Linux tried to open a directory called **etc** within **/etc/ssh**

**6. What type of file object is bin?**

- Regular directory
- Special file
- Link
- Named pipe

**7. The /bin directory typically contains which of the following files?**

- Files marked for deletion.
- Essential command-line utilities.
- Shared program libraries.
- Optional software package files.

**Which of the following directories is a virtual file system (VFS) that represents system information reported by the kernel?**

- /usr
- /proc
- /lib
- /opt

**You've installed a third-party software package called MyApp. Where is the most likely place that this package is stored?**

- /opt/myapp
- /usr/myapp
- /sys/myapp
- /bin/myapp

---

## Activity 4-11: Discussing Storage Issues

---

**1. What does the -h switch in the command df -h do for you?**

It presents disk free (df) data in a human-readable format (megabytes and gigabytes) for mounted file systems.

**2. What does the iostat /dev/sda command do, and how can it be useful in troubleshooting?**

This command generates read and write statistics for the /dev/sda block storage device. It can be useful in troubleshooting because it can help you monitor storage usage statistics and identify poor performance associated with that device.

**3. Which directory or file system is a likely candidate on which to impose storage quotas on a shared Linux system?**

The /home directory or file system because users tend to store everything without regard to space used. Quotas help prevent this kind of storage waste and file sprawl.

**4. As an administrator, you attempt to set quotas on /opt/finance for the finance group, but you receive the error that the mount point /opt/finance is not found or has no quota enabled. You verify that /opt/finance exists. You're trying to enable quotas, so it makes no sense that the error is that the file system has no quotas enabled. What is the problem?**

You didn't edit /etc/fstab to add usrquota,grpquota options to the file system you wish to impose quotas on. You also have to unmount/remount this file system after the change is made to /etc/fstab

**5. Why would an administrator use the ioping utility?**

The ioping utility displays real-time I/O latency data to a specified volume.

---

## Activity 4-12: Tracking Storage Space Usage

---

**2. What command might you issue to track the storage usage of files on the data backup volume?**

Answers may vary, but one example is: sudo du -h /backup/data

**5. How might you use these statistics to diagnose issues with a storage drive?**

Answers may vary, but you can compare the read and write statistics of the drive to see if they do or do not match a baseline of expected behavior. If they don't, the drive may be a bottleneck and causing sluggish system behavior when users and applications read from and write to storage.

**6. What command might you issue to track I/O latency in real-time by sending 100 requests to the data backup volume?**

Answers may vary, but one example is ioping -c 100 /dev/backup/databk

- 7. Which of the following Linux I/O schedulers is optimal in situations where a storage device performs its own I/O sorting operations, or in non-mechanical devices like SSDs and USB flash drives?**

- cfq
- anticipatory
- deadline
- noop

## Activity 4-13: Configuring Storage Quotas

- 7. The user has reached the soft limit for storage blocks. What does this mean as far as the user's ability to write data to this file system?**

The user will be able to continue to exceed this soft limit for a default of seven days. If they go below the soft limit, the timer will reset. If they don't go below the soft limit within the grace period, or if they exceed the hard limit within the grace period, then they will be unable to write to the file system.

## Activity 5-1: Discussing Creating and Editing Text Files

- 1. You're editing a file in Vim and you realize that you've made several errors and decide to start over with your editing. How do you leave the file without saving any changes that you've made?**

Press **Esc** to enter command mode and then enter **:q!**

- 2. You want to create a shell script using Vim. You enter Vim but can't type any letters. What do you need to do before you can begin typing?**

You have to press **i** for insert mode and then begin typing text.

- 3. You've created a script in Vim and note that you haven't entered any instructions or documentation into the file. Standard practice is to add your comments above the line of code that you're describing. How do you open a new line above the one you're currently typing on?**

Press **Esc** to enter command mode and then press **O**.

- 4. How do you copy and paste text in the Vim editor?**

Press **Esc** to enter command mode, then type **y** for yank (copy) and then **p** for paste.

- 5. For text file editing in Linux, you have options other than Vim. Name one graphical editor and one other command-line interface (CLI) editor that you might use instead of Vim.**

You can use gedit for the graphical editor and GNU nano for the CLI editor.

*Solutions*

---

## Activity 5-4: Discussing Searching for Files

---

- You tried using locate to find README files on your new Linux system but quickly realized that it wasn't installed at build time. After installation, you tried using locate to find README files, but again the command has failed to find anything. You discover that for locate to work, you must build the mlocate database. Which command can you run to build the mlocate database?**

`sudo updatedb`

- You tried using locate to find the cat command but locate searches for strings and returned more than 2,000 hits. You need something more efficient for searching for a specific file, especially commands. Which command can you run to selectively find the cat command?**

`which cat`

- You and other junior Linux administrators have noticed that some basic commands don't have man pages associated with them. You would like to know which ones do. One of the other administrators told you about a command that would also list the man pages for a command search. To which command was he referring?**

`whereis`

- The `find` command is one of the most powerful in a system administrator's toolbox. What does the following `find` command do? `find / -name "carbon*" -print`**

This `find` command searches from the root (`/`) directory and all subdirectories for any files beginning with the word "carbon" and their locations.

- Which of the search commands at your disposal would you use to locate a user's lost files, knowing that the user has access to dozens of directories and subdirectories as a member of several organizational groups?**

The `find` command can locate files by owner or group, plus it displays the absolute path location.

---

## Activity 5-5: Searching for Files

---

- What are some advantages of using the `find` command over using the `locate` command?**

The `locate` command requires that a database be updated in order to perform accurate searches, whereas `find` does not. Also, `locate` cannot filter its search by specific directories, whereas `find` can. However, `locate` may be able to perform searches more quickly in certain cases.

---

## Activity 5-6: Discussing Performing Operations on Files and Directories

---

- 1. Linux users use the `cat` command to print a file's contents to the screen, but the original purpose of `cat` is to concatenate or join the contents of two or more files together into a single file. You have files `test.1` and `test.2` that you want to combine into a single file, `test.3`. How can you accomplish this using `cat`?**

`cat test.1 test.2 > test.3`

- 2. As a system administrator, what application of the `tail` command comes to mind first for checking on system status or an application's status?**

Using `tail` on a log file to see the most recent entries should come to mind for checking the status of the system or an application.

- 3. There are two utilities that are good for paging through very long files with search capability. Both enable you to stop, proceed line-by-line, page-by-page, and to page up. Some administrators prefer one utility over the other because it has a few more options. What are these two utilities?**

`more` and `less`

- 4. Your manager asks you to copy all the files in the `/opt` directory to `/backup` for further inspection and analysis. Rather than copying every file and subdirectory manually, how can you copy `/opt` and all its contents to `/backup` with a single command?**

`sudo cp -R /opt /backup` —This command copies the `/opt` directory recursively to `/backup`

- 5. Diane Smith has left the company and you want to preserve her home directory, so you decide to rename it. How can you rename `/home/dsmith` to `/home/dsmith.keep`?**

`mv /home/dsmith /home/dsmith.keep`

---

## Activity 5-7: Reading Files

---

- 4. Why might printing only the first or last few lines be preferable to reading the entire file?**

Answers may vary, as it depends on the purpose and format of the text file. For logs, reading the last 10 lines is a much quicker way to see the latest events on a system than using `less` would be. Printing the first 10 lines might be useful in situations where entries are repeated or otherwise superfluous, and you only need to see a few examples to grasp the idea.

## Activity 5-9: Discussing Processing Text Files

---

1. **The echo command is a very powerful one indeed. It can be used in scripts or interactively at the command-line to do several different things. How can you use the echo command to display your name on the screen?**

`echo "Bob Jones"` —This command sends the output, Bob Jones, to the screen.

2. **How would you use the printf command to display your name and address on different lines using a single command?**

`printf "Bob Jones\n 222 Pine Street"` —The `\n` sends a newline character to output and places the address on a second line. You can get very sophisticated with newline characters and file formatting so that screen text is easier to read.

3. **You have two files that contain more than 100 lines of text each. You need to compare them because, while they're supposed to be similar, checking each one against the other is time-consuming. Fortunately, Linux has a utility to compare the two files. How can you compare file1.txt and file2.txt using this utility?**

`diff file1.txt file2.txt`

4. **A user asks you to help him find a particular string ("audition") in a file, actors.txt. He can't seem to remember which actor he had auditioned for a commercial spot, and he needs to call her back. How can you find his actor using a simple command, rather than read the text file manually?**

`grep audition actors.txt`

5. **You have a file that contains a very long list of activities (activities.txt) that were suggested by coworkers for things to do at the annual picnic. They are in no particular order, but you'd feel better if they were in alphabetical order, making them more readable. How can you rearrange the list into alphabetical order quickly?**

`sort activities.txt`

---

## Activity 5-11: Linking Files

---

6. **Why did the system fail to create the link? What can you do to still create a link?**

You cannot create hard links across different file systems, and the home directory and the backup log directory are on different file systems. To get around this, you must create a soft (symbolic) link.

---

## Activity 5-12: Discussing File Output

---

- How would you create a file with the following content using a single command? Note that each component should be on its own line (a total of three lines), like addressing a letter in the mail:** Bob Jones  
222 Main Street Dallas, TX

```
printf "Bob Jones\n222 Main Street\nDallas, TX"
> bob.txt
```

- Why does the following command send the name Sally Davis to the sally.txt file?** echo "Sally Davis" > sally.txt

Because the `echo` command, coupled with the redirect operator (>), redirects `stdout`, Sally Davis, to the `sally.txt` file.

- Your manager asks you to find the number of times that connections from IP address 10.0.10.5 have connected to server1. These events are typically logged in the /var/log/auth.log file. Using piping, how can you get this count for him?**

```
sudo grep 10.0.10.5 /var/log/auth.log | wc -l
```

This command searches for the string "10.0.10.5" in the authentication log and then pipes the output to the word count utility by the number of lines.

- What does the following command do?** cat file.txt > /dev/null

Anything redirected to `/dev/null` has no output. It isn't saved to a file or displayed to the screen. It is sent to the null device.

- What happens if you issue the following command?**

```
mail bbates@domain.com < addresses.txt
```

The `addresses.txt` file's contents will be mailed to `bbates@domain.com`.

---

## Activity 6-1: Discussing the Linux Kernel

---

- In the monolithic kernel vs. microkernel debate, Linus Torvalds chose a monolithic kernel for Linux. Why?**

Although a monolithic kernel is larger than its microkernel counterpart, the tradeoffs for running system modules, device drivers, and file systems in kernel space, and overall speed, far outweigh the main disadvantage of size.

- As an administrator, it is one of your jobs to keep track of the kernel versions on your Linux systems and attempt to be consistent where possible. Which command can you run to see the kernel version?**

`uname -r` returns the kernel's major and minor version numbers, making it easier to compare versions across many systems.

**3. One of the most important qualities of the Linux kernel is that it is modular.****What does this feature do for Linux as a whole?**

Modularity enables users to configure and extend the kernel's functionality to meet specific needs.

**4. How does the Linux kernel manage devices?**

User space applications send system calls, and the kernel reads the requests and passes them on to the drivers that manage device activities.

**5. How does the Linux kernel handle memory management for applications?**

The kernel maps or allocates the available memory to applications or programs on request and frees the memory automatically when the execution of the programs is complete. This ensures that memory can be allocated to other programs.

---

## Activity 6-2: Exploring the Linux Kernel

---

**2. What is the base version of your currently running kernel according to the `uname` command?**

- 2.4
- 2.6
- 3.4
- 3.10
- 4.18

**3. True or false? According to the `uname` command, you are running a 32-bit hardware platform.**

- True
- False

**4. Which function is associated with the SCI layer of the kernel?**

- Passing requests to device drivers.
- Sending service requests to the kernel.
- Allocating processor time for functions.
- Processing scheduling functions.
- Organizing files on the file system.

**5. What are the major functions performed by the kernel? (Choose two.)**

- Kernel initialization
- Process management
- Memory management
- Module installation
- Dependency management

**6. Which of the following accurately describe the user space? (Choose two.)**

- It is the area of the memory where the kernel executes its services.
- It is the area of memory in which most high-level software runs.
- It is the part of the system that only logged in users can access.
- It is the area of memory in which background processes and low-level system libraries run.

**7. What is one disadvantage of a monolithic kernel compared to a microkernel?**

- Monolithic kernels are slower to access devices.
- Monolithic kernels are larger and consume more RAM.
- Monolithic kernels have a smaller kernel space and are less extensible.
- Monolithic kernels can only run the bare minimum software to qualify as a fully functional OS.

**8. True or false? The Linux kernel is modular, enabling users to extend its functionality.**

- True
- False

---

## Activity 6-3: Discussing Installing and Configuring Kernel Modules

---

**1. You're sure that you compiled and loaded the kernel module for your new Ethernet card, but the card doesn't work. How can you check to be sure the kernel module is loaded?**

Use the `lsmod` command to list all currently loaded kernel modules.

- 2. You have downloaded a new kernel module that you want to load into your running system's kernel. You do so using the `insmod` command. You checked, and the kernel module is in the list of loaded modules, but it isn't working. What might have gone wrong?**

The `insmod` command loads only a specific module and does not load any dependent modules. It's likely that the module you loaded has a dependency.

- 3. You loaded a kernel module that has caused your system to become unstable. How do you remove the errant module?**

Use the `rmmmod` command.

- 4. The developers at your company have created custom kernel modules to optimize in-house application performance. They supply you with all the necessary files to load the new modules, including dependent modules. When you load the first primary module using `modprobe` you receive multiple errors and the modules don't load because of broken dependencies. Which file or files can you examine to find the issues?**

The `modules.dep` file is the likely problem. This file identifies how modules and their dependencies are linked.

- 5. Your manager comes to you to discuss performing some no-cost security hardening on your Linux systems. Specifically, she asks you to protect the kernel from attacks. Which command and associated configuration file can you investigate to assist you in this task?**

The `sysctl` command and its configuration file, `sysctl.conf`, are used to tune and to set security parameters for a running kernel.

## Activity 6-4: Installing and Configuring Kernel Modules

- 3. Do any of these look like they could be a driver for a USB device that can send and receive Bluetooth signals?**

Answers may vary, but `btusb.ko.xz` is the most likely candidate.

- 7. Notice that there are other modules that begin with `bt`, as well as a module called `bluetooth`. Why were these added to the kernel as well?**

These are modules that `btusb` depends on in order to function. The `modprobe` command automatically installs dependent modules when necessary.

## Activity 6-5: Discussing Monitoring Kernel Modules

- 1. You're a new system administrator and you might not know all the commands to find out every detail about a Linux system. Although you don't know the commands, where can you go on the file system to find out the same information in text file format?**

The `/proc` file system contains text files that hold process information, hardware information, kernel parameters, modules, and much more.

2. A junior system administrator asks for your help in determining which components make up the Linux server he's working on. He needs to know brands, models, and other details of certain components to buy replacement parts. To which command or file(s) can you direct him?

He should run the `dmesg` command to see a complete list of all attached hardware components and their details. Optionally, he can look at the `/var/log/dmesg` file.

3. You believe that the network interface card is failing on one of your servers and you need to get the specifics of it so that you can replace it with the same product. Without going to the data center and opening the server's case, how can you find the network card's details?

`dmesg | grep -i network`

4. Your Linux web server seems to have a memory problem that you can't pinpoint through checking web logs or system logs. Where can you look to find details about system memory usage?

The `/proc/meminfo` file contains all the system's memory information.

5. You need to know which file systems are supported by one of your Linux systems. How can you find out?

Look at the `/proc/filesystems` file to see a list of supported file systems.

## Activity 6-6: Monitoring Kernel Modules

1. When was the kernel last compiled?

Answers may vary, but the version used to develop this course was compiled on April 20 of 2018.

### What version of the GCC is your kernel running?

Answers may vary depending on when the kernel was compiled. For the kernel version used to develop this course, the GCC version is 4.8.5.

### Why might this information be useful?

Answers may vary, but validating the kernel version and related information can help you diagnose issues that apply to specific versions, such as incompatible software.

## Activity 7-1: Discussing the Linux Boot Process

1. Your coworkers are trying to decide between using UEFI or legacy BIOS for some of your systems. They ask you for your opinion. What are the advantages of UEFI that you can tell your coworkers?

UEFI is faster, can address a larger amount of memory, can access huge drives, can use more hardware types, and has improved security over BIOS.

2. MBR is to BIOS as \_\_\_\_\_ is to UEFI.

GPT. GPT is part of the UEFI standard and is the successor to MBR.

**3. What is the importance of the initrd phase of the boot process?**

The initrd enables the system to be started in two phases. In the first phase, the system is booted with the minimal set of modules required to load the main, or permanent, root file system. In the second phase, when the main root file system is mounted, the previously mounted initrd file system is removed and the user space boot process continues.

**4. You've taken a new position as a Linux administrator and you discover that every Linux system contains a /boot/efi/ directory. What is the significance of this /boot/efi/ directory?**

Systems with the **/boot/efi/** directory use the UEFI boot process rather than the BIOS boot process.

**5. Linux administrators don't enjoy seeing a kernel panic during the boot process because it means that the system won't recover on its own and something serious has happened. What are some possible causes of a kernel panic that occur during the boot process?**

The system has a corrupted kernel, the **Systemd** program didn't execute, the kernel can't find or mount the root file system, or a hardware component has malfunctioned.

**Activity 7-2: Identifying Linux Boot Components****1. What must happen before the boot loader can run?**

The BIOS/UEFI must be initialized. The processor checks for the BIOS/UEFI firmware and executes it. BIOS/UEFI checks for bootable media and locates a valid device to boot the system. BIOS/UEFI then loads the primary boot loader from the MBR into memory.

**2. When the boot loader completes its tasks, what happens next?**

It transfers control to the kernel.

**3. What is the role of the MBR?**

It contains the partition tables, determines the currently active partition, and determines which partitions are bootable.

**4. Which of the following are advantages that GPT has over MBR? (Choose two.)**

- GPT has larger maximum storage space for partitions.
- GPT enables you to select multiple operating systems to boot from.
- GPT stores boot data in multiple locations for redundancy.
- GPT can support multiple file system types.

**5. What is the purpose of systemd during the boot process?**

It mounts the file system based on the **/etc/fstab** file and begins the process of starting services.

**6. Which boot process component configures device drivers?**

- The boot loader.
- The kernel.
- The inittab file.
- The BIOS/UEFI.

**7. Which of the following boot options enables users to mount a file share as the root file system, rather than storing that file system locally?**

- Boot from NFS
- Boot from ISO
- Preboot Execution Environment (PXE)
- Boot from HTTP/FTP

---

## Activity 7-4: Discussing GRUB 2

---

**1. You want to create a bootable DVD-ROM Linux distribution for some kiosk computers that customers will use to look up products while in your retail locations. Which new GRUB 2 improvement makes this an easier task compared to a few years ago?**

Support for live booting makes it easy to run the OS entirely in memory without installation.

**2. You decided to make all your systems more secure by adding a GRUB 2 boot loader password. After making the change to multiple systems, you feel confident that the new security measure will be effective in securing your systems. Which command can you use to generate a password for GRUB 2?**

grub2-mkpasswd-pbkdf2

**3. You have tasked a junior administrator with adding a single boot loader option to three of your Linux systems. The administrator lets you know that he was unsuccessful at making the changes because he received a permission denied error. What is the issue?**

The administrator must have superuser privileges to create a new GRUB 2 custom menu file.

**4. How can you edit the GRUB 2 boot loader during the boot process?**

You start or reboot the system, then, on the GRUB 2 boot menu press **Esc**. Next, highlight one of the entries, press **e**, edit the configuration, then press **Esc**. Finally, press **Enter** to boot using your option.

## 5. What are some examples of changes you could make to the default GRUB 2 menu?

You could change options such as how many seconds GRUB 2 will wait before automatically selecting the default boot option; whether or not GRUB 2 will order kernel versions in a submenu; whether or not GRUB 2 will display the menu in a graphical terminal; etc.

## Activity 8-1: Discussing Localization Options

### 1. The date command is very powerful, especially for use in scripts. You can add timestamps to log files and to script output. How can you display the day, date, and time using the date command?

`date +"%A %F %X"` —The `date` command by itself also produces similar output.

### 2. You recently moved several Linux server systems from your local data center in the Eastern U.S. time zone to a disaster recovery data center in the Central U.S. time zone. You need to change time zones for those systems. What steps can you take to change the time zones?

Use `sudo timedatectl list-timezones` to find the correct format for the new time zone, which is America/Chicago. Use `sudo timedatectl set-timezone America/Chicago` to set the time. Check the setting with `sudo timedatectl status`

### 3. Your company has deployed physical systems in its data centers rather than virtual ones. Physical systems often experience some time drift if not calibrated with an external source, such as an NTP server. You should set the hardware clock to UTC and also correct any time differences by telling the hardware clock to compensate for drift. How can you set the hardware clock and adjust for drift?

`sudo hwclock -u` sets the clock to UTC and `sudo hwclock --adjust` compensates for drift.

### 4. Your company is global in scope and you have systems in data centers in several different countries. Often while typing you discover that some odd characters appear on the screen. One of your counterparts in one of the offshore countries informs you that they have changed the keymap to better suit their needs. How can you change the keymap for U.S. compatibility?

`localectl set-keymap us` —You can set your keymap as a regular user.

### 5. Why would you need to change the locale settings of a system based on its global location?

A system's locale determines how it will represent various culture-specific elements, the most prominent of which is the language used in the interface. However, a locale can also determine factors such as how date and time are formatted, how monetary values are formatted, and more.

## Activity 8-3: Discussing GUIs

### 1. True or false? A graphical desktop environment is required for most administrative activities, such as creating new users and installing new software.

False. In fact, in most enterprises, the GUI is not installed at all.

- 2. What is the primary architectural difference between the X and Wayland display servers? How does this difference affect performance?**

In X, the compositor is separate from the display server. In Wayland, the compositor and display server are one. The Wayland compositor can therefore interact directly with clients, minimizing latency.

- 3. Where can you, as a user, switch to a different graphical user interface (KDE to GNOME, for example)?**

At the graphical login screen.

- 4. At a foundational level, many remote desktop protocols (e.g., NX, xrdp, SPICE, VNC) are not designed to be highly secure. How can you make them more secure?**

By tunneling through SSH.

- 5. Why would an administrator need or use console redirection?**

Without an operating system on the hardware, an administrator needs some method of connecting to a "bare metal" system. Console redirection is that method.

## Activity 8-5: Discussing Services

- 1. After installing the Apache web server, httpd, you want to check its status. How can you check the status of the httpd service?**

`sudo systemctl status httpd.service`

- 2. How can you activate a service such as httpd?**

`sudo systemctl start httpd.service`

- 3. You want to be sure that the Apache web server starts on boot. Which command can you use to ensure that it starts?**

`sudo systemctl enable httpd.service`

- 4. What is the SysVinit equivalent of systemctl**

`enable httpd.service?`

`sudo chkconfig httpd on`

- 5. You made a change to the /etc/httpd/conf/httpd.conf file and saved it, but the change you made hasn't taken effect. What is the problem and how do you fix it?**

The service needs to be restarted so the configuration file can be re-read and the changes applied: `sudo systemctl restart httpd.service`

---

## Activity 8-7: Discussing Process Issues

---

- One of the other system administrators on your team states that a process is now a zombie. What is the characteristic of a zombie process that makes it a problem for administrators?**

A zombie is a process that was terminated, but has not yet been released by its parent process. It is in a "zombie-like" state where it cannot accept a kill signal because the process isn't available anymore.

- The Linux system that controls your phone system has experienced some problems with memory lately, and you're curious as to what the problem is. Upon further investigation, you note that the system is fully updated and patched. You decide to check running processes to determine if something has gone awry with one of the services. Which command can you use to check running processes?**

`ps -aux`

- You've had several complaints about the performance of one of your systems. You log in and find that it responds slowly to commands. You want to look at processes that might be consuming too many resources. Which command can you use?**

`top`

- While investigating complaints of system "slowness," you use the `top` command, but aren't seeing any issues from a CPU usage standpoint. You want to check memory usage while still using the `top` command. Which command key can you use to display processes by memory usage?**

`M` (uppercase).

- It's 6:00 P.M. and you need to leave for the day. However, you also need to run an audit script that creates a report of all files that haven't been accessed in more than 180 days. If you log off, the script will stop running. How can you accomplish both—running the script and leaving work?**

Run the script using the `nohup` command: `sudo nohup fileaudit.sh`

---

## Activity 8-10: Discussing CPU and Memory Issues

---

- For a quick glance at CPU performance, what does the `uptime` command tell you about the system?**

The `uptime` command displays the CPU's load average for the last 1, 5, and 15 minutes.

- The Performance and Capacity Team in your company requires a system activity report on one of your Linux systems. The team contact has requested a CPU report. Which command can you issue to gather the information for the report?**

The `Sar -P ALL` command gathers CPU statistics for all CPUs.

3. Some of your users have complained about one of the development servers being slow. With what command can you check the memory statistics that give you the best quick snapshot of memory performance?

`free -h -t` gives you a complete look at memory performance in human-readable format. This also includes used swap space, which is important for determining if a system has serious memory issues.

4. The `vmstat` command is useful for displaying an overall performance snapshot, but running it once only displays information since the last reboot. How can you see a more comprehensive view of system performance using the `vmstat` command?

The standard command to see system performance statistics using the `vmstat` command is `vmstat 5 5`—This provides you with a comparison view from last reboot and four updated snapshots.

5. You may configure swap space on a Linux system but not necessarily make it active until you feel like the system requires it. How can you enable a system's configured swap space?

The `sudo swapon -a` command activates all configured swap space on a Linux system.

---

## Activity 9-1: Identifying the Types of Linux Devices

---

1. True or false? Linux supports a unique set of device standards, and devices that work on other operating systems will usually not work with Linux.

True

False

2. What type of device is /dev/sdb?

It is a storage device such as an internal SATA drive or a USB storage device.

3. Some users are concerned that their devices, whether USB, wireless, SCSI, etc., won't be compatible with Linux. What can you do to help ensure that these devices will work with Linux?

Answers may vary, but ensuring that Linux has the correct drivers installed for each device will help ensure that the devices work. Many device drivers come with Linux by default, so users may not even need to configure them manually.

**4. What is the purpose and function of a network adapter?**

- To translate wireless signals into physical data transmissions, and vice-versa.
- To provide an interface for connecting hosts to a network so that they can exchange data with one another.
- To establish a network segment that multiple computers can connect to in order to exchange data with one another.
- To forward transmitted data from one network to another.

**5. True or false? It is common for specialized Linux servers, such as firewalls, to contain more than one network adapter.**

- True
- False

**6. True or false? Printers can use a wide variety of interfaces, and can even be accessed over a network.**

- True
- False

**7. Which of the following hardware interfaces carries audio and video signals and is used to connect devices like monitors?**

- USB
- HDMI
- Wi-Fi
- VGA

**8. Which of the following wireless connection standards is primarily used to create a local area network (LAN) for home and office computing?**

- NFC
- Bluetooth
- Wi-Fi
- RFID

**9. Which of the following is a hardware component that connects a host system to a storage device to facilitate input and output of data?**

- HBA
- SATA
- SCSI
- GPIO

**10. Which of the following is a computer bus standard used by high-performance NVMe solid-state drives (SSDs)?**

- SATA
- SCSI
- PCIe
- SAS

**11. Which of the following hardware connection technologies is primarily used in enterprise storage because of its fast speeds and high reliability?**

- SCSI
- SAS
- PCI
- SATA

---

## Activity 9-2: Discussing Configuring Devices

---

**1. What task can you perform to be able to refer to a device with the same custom name every time it is plugged in?**

Create a symbolic link to the device through a udev rule in /etc/udev/rules.d/

**2. What is the significance of udev?**

Prior to systemd and new kernels, the /dev/ directory only contained static device files—udev brings dynamic device creation and removal.

**3. How can you start the CUPS service?**

`systemctl start cups`

**4. How do you send the file students.txt to the HPLJ5 printer?**

`lpr -P HPLJ5 students.txt`

- 5. There are two device types recognized by the udev device management system: coldpluggable and hotpluggable. When does the system load modules for each device type?**

Coldpluggable device modules are loaded at boot and hotpluggable device modules are loaded dynamically by the kernel when a new device is plugged in during normal operation.

## Activity 9-5: Discussing Monitoring Devices

- 1. You want to check the print queue before you attempt to reboot the system. How do you check the print queue for the non-default printer, HPLJ5?**

`lpq -P HPLJ5`

- 2. What does the `lpq -a` command display?**

It displays print jobs on all configured printers.

- 3. What does the `lsusb` command display?**

`lsusb` displays all USB buses and a list of USB-attached devices and the buses they're connected to. The list will likely be different for every system.

- 4. Which command can you use to display information about all block devices and how can you find out more information about each device in the list?**

Use `lsblk` to list all block devices and their parent devices. Use `udevadm info` to display specific information about each device. The `dmesg` command also provides some device details.

- 5. From where does the `lsdev` command pull its displayed information?**

It draws its information from the `/proc/interrupts`, `/proc/ioports`, and `/proc/dma` files.

## Activity 9-7: Troubleshooting Hardware Issues

- 1. What is a common source of problems with keyboards, printers, and storage adapters?**

Missing or poorly configured drivers are a common source of hardware problems.

- 2. What is the term used to describe the situation where a process fails to free up allocated memory when it is no longer needed?**

This is referred to as a memory leak.

- 3. What are two common hardware-related issues that are not a result of faulty drivers or insufficient memory?**

Loose or damaged cables and improperly slotted or plugged devices.

**4. Which of the following commands is an often unreliable way of querying the system for attached hardware components?**

- lshw
- vmstat
- free
- dmidecode

**5. Which of the following connection protocols may lshw *not* detect?**

- USB
- FireWire
- SCSI
- PCI

**6. A user is complaining that their USB mouse is unresponsive. Which of the following commands can you use to quickly pull up information about the device, including its vendor, product model, and drivers?**

- lshw -c input
- lshw -c mouse
- dmidecode -t input
- dmidecode -t mouse

**7. When a user presses the Shift+2 on their keyboard, they expect the @ symbol to be entered, but instead, the # symbol is. What can you do to troubleshoot this issue?**

Answers may vary, but the most useful approach to take is to examine the language and keyboard mapping settings on the computer. It's likely that the wrong keyboard format is set—to another language, region, or keyboard style—causing unexpected characters to be entered upon certain keystrokes.

**8. Several users have been issuing multiple print jobs to the printer, causing it to become backed up and unresponsive. You enter the lpq command to query the print jobs, and notice that one job is very large and likely causing the bottleneck. Which of the following commands should you enter to stop the offending print job and free up the queue?**

- lpq -s {job number}
- lprm {job number}
- lpq -s {print file name}
- lprm {print file name}

- 9. You recently installed a new graphics card in one of your Linux systems that will be used by Develetech's video editor. The video editor notices sluggish performance and repeated crashes in their video editing software. You've deduced that the software isn't the source of the problem; the graphics card is. What step(s) should you take to try to solve this problem?**

Answers may vary, but when it comes to graphics card issues, one of the most important troubleshooting steps is to ensure that you always download the latest drivers available from the manufacturer. These drivers often fix performance and stability issues, especially when the graphics card is put to use by video-intensive software.

- 10. An administrator has reported that a Linux server is sluggish, unresponsive, and frequently triggers a kernel panic with the error message "Machine Check Exception". What can you do to diagnose and fix this issue?**

Answers may vary, but this particular error usually indicates faulty hardware. You can redirect the exact error code into the `mcelog` command to get more information about the error. Since the system is sluggish and unresponsive, you might see an error-correcting code (ECC) error, indicating that one of the memory modules has failed. You can then run a program like `memtester` to stress test your RAM modules.

- 11. One of Develetech's software developers has been programming on a Raspberry Pi device. He's been trying to connect the Pi's serial cable to a Linux system so that he can see the output of the program on the Linux console. However, the output doesn't appear. What can you do to troubleshoot this issue?**

Answers will vary. You should perform standard steps like ensuring the correct drivers are installed, checking the physical cables for damage and to see if they're properly slotted into the serial port, etc. If this doesn't resolve the problem, you should make sure that the serial device is using the expected serial console, typically located at `/dev/ttyS#` where `#` is the number of the console (starting with 0). You may need to configure the Pi's settings so that it uses this console by default. Also, by default, only the root user is allowed to access serial ports. If necessary, use `chmod` to grant access to `/dev/ttyS#` for the appropriate accounts.

## Activity 10-1: Identifying TCP/IP Fundamentals

- 1. List at least five application-layer protocols.**

Examples include: HTTP, HTTPS, SMTP, SSH, Telnet, FTP, TFTP, SNMP, DNS, DHCP, etc.

- 2. What are the two transport-layer protocols?**

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)

- 3. What protocol within the TCP/IP stack is responsible for logical addressing?**

Internet Protocol (IP)

- 4. What are the names and numbers of each layer of the OSI model? What is each layer's function?**

Layer 7 (application) supports software and users; Layer 6 (presentation) formats data for use; Layer 5 (session) manages connections; Layer 4 (transport) manages reliable transmission of data; Layer 3 (network) manages logical addressing; Layer 2 (data link) manages physical addressing; Layer 1 (physical) enables physical connectivity.

- 5. What is the name of the physical address used in Ethernet networking?**

Media access control (MAC) address, permanently set on the network interface card (NIC) by the manufacturer.

**6. What is the name of the logical address used by the TCP/IP protocol suite?**

IP address

**7. What is the human-readable name of a network device?**

Hostname

**8. What are the two components of an IP address?**

Network ID and host ID

**9. What is the bit length of IPv4 addresses?**

IPv4 addresses are 32 bits in length.

**10. List the address ranges for Class A, Class B, and Class C IP addresses.**

Class A = 0.0.0.0–127.0.0.0; Class B = 128.0.0.0–191.255.0.0; Class C = 192.0.0.0–223.255.255.0

**11. List the default subnet mask for Class A, Class B, and Class C IP addresses:**

Class A = 255.0.0.0 or /8; Class B = 255.255.0.0 or /16; Class C = 255.255.255.0 or /24

**12. What is the IP address associated with the loopback address?**

127.0.0.1

**13. What IP address range is associated with link-local/APIPA addresses?**

169.254.x.x

**14. What are two reasons why a network administrator might segment a network?**

Answers may vary, but common reasons are to increase security by grouping hosts with similar security requirements together; and to increase performance through more efficient traffic management.

---

## Activity 10-2: Identifying Linux Server Roles

---

**1. You want to provide secure connectivity to users in your company to a shared server for web and shell-based command-line access. Which protocols will you choose?**

SSH for shell/command-line and HTTPS for web because they are both secure and standard protocols.

**2. Your Windows users complain that there is no native SSH client in their older version of Windows. What can you recommend?**

PuTTY is a commonly used SSH client for Windows.

**3. You need to provide shared Linux resources to Windows users. Which service can you set up on your Linux system to accomplish this?**

You can set up Samba to create Windows-type directory shares.

**4. Email is an essential network service. Which programs might you select for setting up email services on your Linux server?**

Sendmail or Postfix are common Linux email services.

**5. Mobile devices have created a mobile workforce. Your users need to be able to connect to the corporate network to access shares, printers, and other internal tools. Which service can you set up to enable secure connectivity for your mobile users?**

A virtual private network (VPN) service works well because it encrypts all data between the user's device and the corporate network.

**6. Rose Stanley, a graphic designer at Develetech, has asked for your help. She regularly transfers many very large image files. She wants an efficient way of doing this over the network. What network service do you recommend?**

Answers may vary, but FTP is a good choice because it is designed for simple file transferring within a client/server architecture.

**7. Chris Mason, a sales executive at Develetech, has been told to connect to a particular server to access resources. He has asked you for help because the instructions indicate he should connect by IP address. He cannot easily remember the IP address and has asked you if there is an easier way to connect to the server. What service can you use to make the connection easier for Chris?**

Answers may vary, but DNS is a good choice because it centralizes and streamlines the name resolution process.

**8. Develetech employees have contacted the help desk because they cannot connect to a particular server. Part of your investigation indicates the server has been configured with a dynamic IP address. How could this contribute to the problem and what steps should be taken to address it?**

Dynamic IPs could change, meaning the server's identity on the network has changed, keeping users from connecting to it. A solution is to use a static IP address configuration.

**9. Develetech's security team requires that log files be archived for future reference and audits. Since you are responsible for several Linux servers, how can you make this process more efficient?**

Answers may vary, but you can centralize the log files on a single Linux server.

**10. One of the developers at Develetech has asked for your help. She needs a Linux test environment to verify her application functions as designed. She would like to manage it herself and be able to revert it back to its original configuration after each test. What solution can you suggest?**

Answers may vary, but creating a virtual machine on a platform like KVM is a good choice because it is native to the Linux kernel.

## Activity 10-4: Discussing Connecting to a Network

**1. You are setting up a new Linux server on your corporate network. For a computer to participate on a network, what must the server possess?**

To participate on the network, the server must have a valid identity as well as know the location of a few key services.

- 2. Your manager has set up a new project for you and your team to give descriptive hostnames to all Linux servers. How can you set a system's hostname to fswebserver01?**

`sudo hostnamectl set-hostname fswebserver01`

- 3. You ask a coworker to check the IP address on server03. He attempts to use ifconfig but it isn't installed. How can you instruct him to check the IP address?**

Tell him to enter `ip addr show`

- 4. One of the administrators on your team prefers working with graphical user interfaces (GUIs) rather than at the command-line. However, there are no GUI elements installed on your Linux servers. He needs to configure some additional networking elements for several systems. Which command can you advise him to use?**

The `nmtui` program is a text-based user interface to NetworkManager that is used at the command-line, but also provides a simplified interface.

- 5. On Linux systems that have a GUI installed, you configure network settings using a NetworkManager graphical utility. What is the name of that utility?**

The graphical user tool for NetworkManager is `nmgui`

## Activity 10-6: Discussing DHCP and DNS Client Services

- 1. You have set up two internal DNS servers in your network and need to check if they're working to resolve hostnames to IP addresses and IP addresses to hostnames. Which utility can you use to do this?**

Any of the name resolution utilities will work: `nslookup`, `host`, or `dig`

- 2. Your team set up a VPN connection between the main corporate office and one of your branch offices, but you cannot connect to any system in the branch office network. One of the servers at the branch office has an IP address of 192.168.1.10. Which command can you use to check the network path between your computer and that server to help troubleshoot the problem?**

The `traceroute` command is useful in checking such a connection:

`traceroute 192.168.1.10`

- 3. You installed the Apache web service on Linux server web01, but you can't connect to the web service (TCP port 80) from another computer. To begin troubleshooting, you use a utility to check TCP connections and ports that web01 is listening on. Which utility can you use?**

Use the `ss -l` command to display a list of listening sockets to determine if the web service is listening on port 80.

- 4. One of the other administrators on your team patched and rebooted a server, but is worried because it is taking a long time for the server to become available again. Which utility can you use to check when the system comes back online?**

Use the ping command to check when the server is available on the network again.

- 5. You suspect that someone is attempting to hack into one of your servers, so you decide to capture a few minutes of network traffic on that system to investigate further. Which command-line utility can you use to capture the traffic?**

Use the tcpdump command to capture the traffic for offline analysis.

## Activity 10-8: Discussing Cloud and Virtualization Technologies

- 1. Your CIO has tasked you with the job of finding an appropriate cloud service to host some of your applications. The application needs to appear as if it is locally installed. Which cloud model will you select?**

The Software as a Service (SaaS) offering is the appropriate choice because your CIO has tasked you with the definition of SaaS itself—applications are hosted, but behave as if they are local.

- 2. Executive management has decided to move toward a more agile infrastructure with mobile users and 100 percent hosted applications and infrastructure. This option enables employees to work from any location and relieves the company of hardware lifecycle and application support. Which option has the executive management decided to move to?**

Public cloud offerings provide companies with off-premise infrastructure with some control of resources, but without the responsibility of maintaining the hardware and software support.

- 3. To move to a public cloud offering for your company's applications, you respond with the recommendation of using one of the two primary cloud service provider (CSP) companies. What are the two primary CSP companies from which to choose?**

Amazon (Amazon Web Services) and Microsoft (Microsoft Azure).

- 4. Your company needs to transition from individuals running virtual machines locally on workstations to deploying virtual machines company-wide. Your team determines that it is more cost-effective for you to use VMware ESXi. VMware ESXi is which type of hypervisor? What does this mean?**

VMware ESXi is a type 1 hypervisor. This means that the hypervisor runs on "bare metal" rather than on top of an existing operating system.

- 5. There is much debate among your team concerning whether to deploy virtual machines with thick-provisioned storage or thin-provisioned storage. What the debate boils down to is performance. Which provisioning type has better performance?**

Thick provisioning has better performance because all space is allocated at configuration time and the storage capacity is fixed rather than dynamic. Fixed capacity has better performance because there is no wait for expansion in the case of thin-provisioned storage.

---

## Activity 10-9: Configuring Virtualization

---

**1. What are some of the potential benefits of virtualization?**

Answers will vary. Virtualization can enable easy-to-revert environments; enable more efficient use of hardware; support on-demand availability; support quick starting and stopping of environments; offer better support for disaster recovery; and more.

---

## Activity 10-10: Discussing Networking Issues

---

**1. You're asked to design the network service implementation at a branch office location. There will be servers, printers, and workstations in the office. Which approach for assigning IP addresses to workstations would you take?**

Dynamic IP address configurations are usually appropriate for client machines (workstations). They may also be used for network print devices.

**2. Your team has decided to use only one internal DNS server and one external. You must change the DNS server IP addresses on every Linux server. Which file can you edit to change the IP addresses?**

The `/etc/resolv.conf` file contains the IP addresses of DNS servers.

**3. Why do we use a service such as DNS to map IP addresses to names?**

Because IP addresses are difficult to remember, and names are easier for humans to remember.

**4. You're assisting a group of web developers to set up their test systems, but you don't want other users to know about them, so you aren't registering them with your network DNS servers. Instead, you're helping the developers edit a local file on their systems that will act as a local IP address-to-name mapper. Which file can you edit for this private name resolution service?**

Edit the `/etc/hosts` file on each developer's workstation to map the test systems' IP addresses to names.

**5. Your team is setting a standard for the `/etc/nsswitch.conf` file. After some debate, the team decides to use the preferred configuration for the ordering of name lookup services. What order is this?**

The preferred configuration is `/etc/hosts` and then DNS.

## Activity 10-12: Troubleshooting Networking Issues

- Rose Stanley calls the help desk, registering a ticket that her Linux workstation does not have network connectivity. What process and tools might you suggest?**

Answers may vary, but could include: a) Confirm the network cable is plugged in properly; b) Use the `ip addr` command to verify whether she has a valid IP address configuration. If the 169.254.x.x IP address is returned, her machine is not able to lease an IP address from the DHCP server; c) Attempt to ping one of the servers on the network that Rose might connect to. If the ping returns a "destination host unreachable" message, there is likely a configuration error on her Linux workstation; and d) Use `traceroute` to attempt to connect to a server. If the `traceroute` fails at a particular "hop" or router along the path, there is likely a configuration issue at that router.

- A change has been made to the configuration of a router by a member of the network team. Unfortunately, the change was not documented and the team member is unavailable. You have no access to the router itself. You would like to investigate what network traffic is moving on each side of the router to help determine what ports might have been closed. What tools might you use to gather network traffic on each side of the router?**

Answers may vary, but could include tools like `tcpdump` and Wireshark.

- What steps might you take to implement a plan of action regarding the gathering of network traffic?**

Answers will vary, but might include: a) Use a packet sniffer like Wireshark to observe what traffic is destined for the router on one segment; and b) Use a packet sniffer on the other segment to observe what traffic has been passed through the router, and what traffic has not. You can begin to infer what traffic is being blocked by the router with this information.

- The Develetech IT staff recognizes the importance of solid documentation and wants to generate a representation of the entire network. The staff wants the information to include all routers and servers, as well as listening services and operating system information for each server. What tool might you suggest for this project?**

Answers may vary, but `nmap` is one of the most useful tools for this job.

- A junior Develetech server administrator believes that the bandwidth usage on his Linux server is being consumed by a particular network service. What tool could you suggest to him to gather information about exactly what protocols are using bandwidth on the NIC?**

Answers may vary, but `iftop` is one of the most useful tools for this job.

- After the administrator gathers the desired information, he discovers that there is no unexpected bandwidth usage by the services. He wonders if perhaps there is an issue on the network itself. He asks for a utility that would enable him to empirically test network bandwidth between two servers. What might you suggest?**

Answers may vary, but `iperf` is one of the most useful tools for this job.

- 
- 7. One of Develetech's server administrators is having difficulties with name resolution. She is responsible for both Linux and Windows servers. She wants to know if there is a single tool she can run from both of her servers to confirm they are receiving name resolution information from a specific DNS server.**

Answers may vary, but nslookup is one of the most useful tools for this job.

---

## Activity 11-1: Discussing Package Managers

---

- 1. Your company has standardized on CentOS as the corporate Linux distribution. You want the greatest amount of control over how programs are installed, where they are installed, and all install options. Which install method works best for your requirements?**

Installing from source gives administrators the greatest amount of control over options at compile time.

- 2. You need to install the Apache web service on your new CentOS web server farm and you cannot decide between using a package manager and compiling. You look at both options thoroughly and decide that, for your current needs, installing via package manager will be adequate. Which package manager do you use to install Apache?**

CentOS is a Red Hat Enterprise Linux derivative and YUM is the preferred package manager to use.

- 3. Your development team at Develetech needs to modify some MySQL database software to optimize it for a database application. You have installed MySQL using YUM. What do you need to do to accommodate the modification need?**

Uninstall MySQL using YUM. Have the developers acquire the MySQL source code, modify it, and then you can compile and install the software manually.

- 4. What is the purpose behind using a package manager to install software?**

Package managers make it much easier to install software, control what software is installed, manage software versions, and uninstall the software.

- 5. You have used Red Hat Enterprise Linux and some of its derivative distributions, but you want to try a Debian-based distribution because of the APT package manager. Aside from Debian itself, what are some common Debian-based distributions you can choose from?**

Ubuntu, Linux Mint, Kali Linux, SteamOS, and openSUSE are all common Debian-based distributions available to you that use APT for software installation.

---

## Activity 11-2: Identifying Package Managers

---

- 1. What is the common package manager and package management utility for Red Hat-derived distributions?**

RPM and YUM, respectively.

**2. What is the common package manager and package management utility for Debian-derived distributions?**

dpkg and APT, respectively.

**3. CentOS Web Server**

RPM, as CentOS is derived from Red Hat Linux.

**Ubuntu Development Workstation**

dpkg, as Ubuntu is derived from Debian.

**Raspberry Pi with Raspbian Linux**

dpkg, as Raspbian is derived from Debian.

**Scientific Linux**

RPM, as Scientific Linux is derived from Red Hat Linux.

**4. Why might you choose to compile software from source code as opposed to using a pre-compiled package?**

Answers may vary, but can include: The software may still be in development, so the package has not yet been created; there are no plans to create a package of the software at all; compiling from source code can support greater customization and modification of the software; and more.

---

## Activity 11-3: Discussing Managing Software with RPM and YUM

---

**1. You are teaching a junior administrator how to install software via YUM. You ask him to install the Lynx text-based web browser (`lynx`) on a test system. What should the install command look like?**

`sudo yum install lynx`

**2. You need to remove the Apache web service software package (`httpd`) so that you can install via source code. Using YUM, what command can you issue to uninstall Apache?**

`sudo yum remove httpd`

**3. You want to install the `xterm` package onto your Linux workstation and don't want to wait for `xterm` and all its dependencies to be found before you acknowledge the installation. Using YUM, how can you install `xterm` and its dependencies without having to interact with the installation?**

`sudo yum -y install xterm`

**4. You need to install several packages that the developers have downloaded and placed into a shared directory (`/scratch`). These are all RPM packages. How can you install the `rdesktop-0.9.rpm` package, for example?**

`sudo yum localinstall /scratch/rdesktop-0.9.rpm`

5. You have taken a position as a system administrator at a new company and have done some discovery work on the environment. You've noticed that multiple software packages are more than one year out of date on several CentOS systems. Using YUM, which command can you issue to update all software packages on each system without acknowledging or confirming the update?

`sudo yum -y update`

---

## Activity 11-5: Discussing Managing Software with dpkg and APT

---

1. You are teaching a CentOS Linux administrator how to install software on a Debian-based Linux system via APT. Which command can you use to install the text-based browser Lynx (`lynx`)?

`sudo apt install lynx`

2. You need to remove the MySQL software package (`mysql`) so that you can install the program via source code, but you want to keep all configuration files intact. Using APT, what command can you issue to uninstall MySQL while leaving the configuration files?

`sudo apt remove mysql`

3. It has been a few months since the old system administrators, who are now no longer with the company, updated the Linux systems. The first thing you need to do is to update the APT database with a fresh list of available packages. How do you update the APT database?

`sudo apt update`

4. You need to totally remove the `emacs` package and its configuration files as well. Using APT, what command can you use to do this?

`sudo apt purge emacs`

5. You have taken a position as system administrator at a new company and have done some discovery work on the environment. You've noticed that multiple software packages are more than one year out of date on several Debian systems. Which command can you issue to update all software packages on each system?

`sudo apt upgrade`

---

## Activity 11-7: Discussing Configuring Repositories

---

1. What is the purpose of creating a centralized internal repository?

This centralized approach makes version control much simpler.

- 2. You have recently taken over support for a group of CentOS Linux servers formerly managed by a different system administrator. You have logged onto each one to check configurations and to update each system. When you attempted to update the systems, no updates were available. Which command can you run to find out which software repositories are configured?**

`yum repolist`

- 3. You have created an internal repository for all your CentOS Linux servers but you are worried that it will be out of date in a few weeks. Downloading a new set of packages every few weeks is a daunting task. Is there any way to automate this process? If so, how?**

Yes, the `reposync` utility updates the repository automatically.

- 4. You have a few Debian-based Linux systems mixed in with your CentOS ones and you need to configure repositories for those. After adding a new repository to the `/etc/apt/sources.list` file, which command can you run to let APT know about the new repositories?**

Run the `apt update` command.

- 5. In a YUM repository configuration, what does the `gpgcheck=0` entry do?**

It disables GPG checking for the repository.

## Activity 11-9: Discussing Acquiring Software

- 1. You need to transfer an entire directory, /2018, containing hundreds of files to five other servers. How can you bundle the files into a single file for transfer?**

`tar -cvf 2018.financials.tar /2018`

- 2. You have a tarball of a program that is 300 MB in size and that is too large for most email programs to handle. You need to somehow make the group of files smaller, but you also don't want to create multiple files to send. What is the solution?**

Use a compression program such as `Gzip` to compress the files so that the collection of files is smaller and more manageable.

- 3. If you downloaded the file `wordsmith-12.3.tar.gz`, how would you retrieve its contents?**

First, uncompress the file: `gzip -d wordsmith-12.3.tar.gz` and then use the `tar` command to extract the files: `tar -xvf wordsmith-12.3.tar`

- 4. How do you use the `wget` command to download software?**

You need to know the complete path to the file you want to download. For example: `wget http://www.site.example/downloads/documents.1.tar.gz`

- 5. True or false? You can use the `wget` command to upload files to a web directory.**

False. The `wget` command is only for downloading files.

*Solutions*

---

## Activity 11-11: Discussing Building Software from Source Code

---

**1. What does the `./configure` command do?**

It gathers system information needed by the application to be compiled and stores it in the makefile.

**2. You have downloaded, decompressed, and extracted a software program tarball, and run the `./configure` command. What is the next step in the installation process?**

The next step is to run the `make` command.

**3. Which command actually compiles the source code into executable code?**

The `make` command compiles the source code using the information in the makefile.

**4. What is the final step in compiling software?**

The `make install` command is the final step.

**5. What does the `make install` command do in the software compilation process?**

The `make install` command copies the compiled binaries to the predetermined directories.

---

## Activity 11-13: Discussing Software Dependency Issues

---

**1. What is a software dependency?**

A software dependency is one or more software packages that must be installed before a desired package is installed.

**2. You have a CentOS system and you need to install an application that you know has a lot of dependencies. What can you do to satisfy those dependencies and install the application?**

Use the `yum install` command to install the application and its dependencies automatically.

**3. You have a CentOS system that runs an application for your company, but anyone who runs it receives errors, as if some critical parts of the application were missing. How can you verify the application's components?**

Use the `rpm -V <application name>` command to verify that all components of the application are installed.

- 4. Being security conscious, you always check out any software package being installed to your CentOS systems. Which command can you run to check supporting software packages prior to installing a new software application?**

The `yum deplist` command displays dependencies so that you can see what's going to be installed with an application before it's installed.

- 5. Why do you need to assume root privileges when running the `make install` command?**

Because the `make install` command writes to system directories that only root has permission to write to.

## Activity 11-14: Troubleshooting Dependency Issues

- 1. In what way are yum and apt more effective at managing software dependencies than rpm or dpkg?**

Answers may vary, but `yum` and `apt` can automatically install dependency software, assuming that software is available in the repository.

- 2. Michael Anderson, a Develetech employee, calls you for help with software. He believes some files may have been deleted from his CentOS system, and now a piece of software he needs does not run. What command could you run to check that all the necessary components of the software are installed on the system?**

`rpm -V <package name>`

- 3. Andrew Riley, a Develetech Marketing department employee, has been given privileges for installing software on his Ubuntu laptop. He has downloaded a package and is using the `rpm -ivh <package name.rpm>` command to install the software. He says the system error indicates that the `rpm` command is not found. What is the issue?**

The `rpm` command is not available by default on Ubuntu, and is instead found on Red Hat-derived distros. Andrew should be using `apt` instead.

## Activity 12-1: Discussing Cybersecurity Best Practices

- 1. True or false? If one of the CIA triad principles is compromised, the security of the organization is threatened.**

True.

- 2. Which authentication method would you use if you didn't want to issue hardware tokens, install special software, or require passwords of your users?**

Biometrics is one such authentication method. The user's physical characteristics, such as retina, fingerprint, or voice, are used in place of passwords, tokens, or specialized software.

- 3. Why is multi-factor authentication effective in securing network and computing assets?**

Because it doesn't simply rely on one factor of authentication, but adds another external knowledge item, token, or attribute that is extremely difficult to counterfeit.

**4. What is the purpose of using a chroot jail?**

The chroot jail isolates or confines a process or program to its own directory structure. The process cannot live or access anything beyond the isolated chroot location (directory).

**5. Why is encryption essential to secure communications on a system or over the network?**

Because only someone or some process with the proper key can decrypt the information, which makes captured information useless to anyone without the proper key.

---

## Activity 12-3: Discussing IAM Methods

---

**1. You and the security team decide that, to enhance the security at Develetech, you need to use something more secure than passwords for your SSH sessions. What other authentication method can you employ to increase security?**

You can set up public-key cryptography.

**2. You notice several files in your home directory under the .ssh directory. One of them is id\_rsa.pub. What is this file's purpose?**

The id\_rsa.pub file contains your SSH public key.

**3. You want to set up SSH keys to better secure your SSH communications between your Linux desktop system and your Linux servers. Which command can you use to generate the public/private key pair?**

The ssh-keygen command generates the public/private key pair.

**4. OpenSSL is standard on Linux systems and is useful for securing data. What are some of its common uses?**

Common uses include: generating public and private keys; generating self-signed digital certificates in various formats; generating digital certificates for other entities based on CSRs; calculating hash values using various functions; encrypting and decrypting data using various algorithms; and managing keys and certificates in a CA.

**5. You need to set up a VPN so that your users can securely connect to the corporate network on public Wi-Fi connections, from home networks, and from any location where they have Internet access. Which software can you investigate that supports password, certificate, and smart card authentication methods?**

OpenVPN is a good choice for further investigation.

---

## Activity 12-5: Discussing SELinux and AppArmor

---

- You are in the testing phase of implementing some new services and SELinux keeps blocking external users from testing. How can you allow access without disabling SELinux or spending a lot of time setting up exceptions to allow all the ports required for testing?**

Place SELinux in permissive mode—this leaves policies in place, but they are not enforced. As violations occur, they are logged.

- You SSH into a Linux system that only allows SSH access. The system is a web server and you've checked to see everything is working properly. However, no one can connect to the web server with a web browser. You suspect SELinux is in enforcing mode and blocking access. How can you check?**

Run the `sestatus` command.

- You are researching a possible breach attempt on one of your SELinux-protected systems. Which command can you use to check the SELinux audit log?**

Either the `sealert` command or the `audit2why` command will provide you with results.

- You are deploying four Debian-based Linux servers and you do not want to use SELinux for access control. What is the most common alternative to SELinux for Debian-based systems?**

AppArmor is the most commonly used alternative to SELinux for Debian-based systems.

- You want to set some very restrictive protection on certain executables. AppArmor gives you that capability. Which mode do you use for the most restrictive profiles for your executables?**

Use enforce mode because it logs violations and prevents any violations from occurring.

---

## Activity 12-7: Discussing Firewalls

---

- What type of firewall examines network packets in isolation, rather than in the context of other packets in a transmission?**

Stateless (packet filtering)

- Several of your users want to install and use Linux at home but are confused by firewall rules and managing a firewall for themselves. What tool can you recommend to help them configure their firewalls?**

The Uncomplicated Firewall (UFW) is a tool primarily useful for home users who don't have the experience with the intricacies of firewall configuration.

- True or false? The Uncomplicated Firewall (UFW) originated with Ubuntu but can be downloaded and installed on other distributions.**

True.

- 4. Your parent company purchased another smaller company that has six Linux systems in its server room, all of which are now your responsibility to manage. They all run the firewalld service. Which command can you use to manage the firewalld service and its associated zones?**

The `firewall-cmd` command enables you to configure `firewalld` by querying, adding, modifying, and deleting zones and services as desired.

- 5. True or false? IP forwarding is the Linux kernel implementation of network routing functionality.**

True.

## Activity 12-8: Configuring a Firewall

- 7. What is the Linux kernel framework that most firewall services rely on to some degree?**

Netfilter

- 8. In an iptables table, what is the function of a chain, and how do chains interact with one another?**

Chains are sets of rules that the table uses to implement firewall functionality. Chains enable a progression of how firewall rules are evaluated; traffic matching a rule in one chain can be passed to another chain, where it is evaluated against a new set of rules.

- 9. True or false? IP forwarding is most useful on systems with only one network interface.**

True

False

- 10. What are the differences between a firewall and an intrusion prevention system (IPS)?**

Answers may vary, but firewalls typically allow or deny traffic into and out of a network based on a set of predefined rules. An IPS evaluates traffic that has made it past the firewall and looks for a repeated pattern of anomalous or otherwise unwanted behavior. An IPS is therefore a second layer of defense. Firewalls can also apply to outgoing traffic, whereas IPSs tend to focus on just incoming traffic.

## Activity 12-9: Discussing Logging Services

- 1. You have set up a training session for new Linux system administrators and you are discussing log files, their format, their location, and their contents. Generally speaking, where are most Linux system logs located?**

Linux system logs are located in the `/var/log` directory.

- 2. You are giving a presentation to the IT and Security teams to convince them that you should adopt a new corporate standard for syslog services on your Linux systems. Your argument is that rsyslogd has several advantages or improvements over the standard syslogd service. What are some of those improvements?**

Improvements include: TCP instead of UDP as the transport protocol; increasing the reliability of transmitted data; data encryption using SSL/TLS; outputting data to various database technologies like MySQL; buffering data on local systems when the remote receiver is not ready to accept it; filtering data based on content; and the **rsyslogd** log file format is backward compatible with the **syslogd** file format.

- 3. Some files were deleted from a shared directory after regular business hours last night and the other users want to know who was responsible for the deletions. To find out who was on the system during the suspected times, you use the last command. From which log file does the last command read its data?**

The **last** command retrieves information from the **/var/log/wtmp** file.

- 4. You suspect that some users have not logged into the file server to update their passwords since the new file server went into production. Which command lists all users and the last time they logged in?**

The **lastlog** command displays the list of users and the last time they logged into the system.

- 5. You need to set up a log rotation schedule that's relatively maintenance-free. In other words, you don't want to have to purge logs from 30 servers on a regular basis—you want it to happen automatically. Which utility provides this service on a schedule that you determine?**

The **logrotate** utility is used to perform automatic rotation of logs.

## Activity 12-15: Discussing Backing Up, Restoring, and Verifying Data

- 1. You and your team have devised a backup plan that includes a full backup once a month and then incremental backups between full backups. What are the two primary advantages of this backup scheme?**

It takes less time to perform incremental backups and it requires less space for the backups.

- 2. Your team determines that for certain systems, snapshots are the appropriate backup method for critical storage contents. What is the advantage of a snapshot?**

Snapshots can be taken as checkpoints that you can restore to a particular point in time if required.

- 3. What is the advantage of using an off-site backup and how can you transfer files from the main site to an off-site facility?**

The advantage is that, in case of a disaster, the data at the off-site facility is probably safe. You can transfer the files by VPN or by using physical media.

- 4. There are several automated processes that copy files between servers for backups. These processes use the standard File Transfer Protocol (FTP). What is wrong about using FTP, and what is the better alternative?**

FTP data is transferred in plaintext (no encryption). The better alternative is Secure File Transfer Protocol (SFTP), which is encrypted.

- 5. A senior-level administrator performed an audit on your backups and commended you for using SFTP to encrypt data as it traverses the network and the Internet. However, he introduces you to a new command, rsync, that has a particularly interesting advantage over SFTP. What is that advantage?**

The rsync command can copy files over SSH, but it also synchronizes (copies differences between) files, rather than copying the entire contents of a directory. Therefore, it saves network bandwidth.

## Activity 13-1: Discussing the Bash Shell Environment

- 1. You are exploring your local (shell) variables on each system that you manage. The first variable you check is the SHELL variable. How can you check the SHELL variable and what do you expect it to display?**

Check the SHELL variable with echo \$SHELL and expect /bin/bash as the output.

- 2. You have seen other administrators use the export command and you have seen it referred to in documentation for scripts. What does the export command do?**

The export command changes a local (shell) variable into a global variable.

- 3. You log into a system that seems to have something incorrect in its configuration, but you are having trouble pinpointing why certain paths don't exist and why certain shell variables are set to non-standard values. How can you view all shell variables at once?**

Use the set command with no arguments.

- 4. You believe either that the PATH variable is incorrectly set or that there are a significant number of commands missing from the system. How can you display the PATH variable value?**

Use the echo \$PATH command.

- 5. Another system administrator has aliased the ls command to ls -la and you don't like to see that much output in a simple ls command. How can you temporarily remove the alias?**

Use the unalias command: unalias ls

---

## Activity 13-3: Identifying Scripting and Programming Fundamentals

---

**1. Which of the following is the correct way to assign a variable in Bash scripting?**

- my\_name=Mary
- my\_name = 'Mary'
- my\_name Mary

my\_name= 'Mary'

**2. Which of the following is the correct way to substitute a variable in Bash scripting?**

- echo my\_name
- echo \$my\_name
- echo 'my\_name'
- echo (my\_name)

**3. True or false? Arrays start with the index 1**

- True
- False

**4. Which of the following is the correct way to substitute array index 2 in Bash scripting?**

- echo my\_arr[2]
- echo \$my\_arr[2]
- echo my\_arr{2}
- echo \${my\_arr[2]}

**5. Choose which of the following code statements accurately produces the following output: The total is \$50.**

- echo The total is \$50.
- echo "The total is \\$50."
- echo 'The total is \\$50.'
- echo "The total is \$50."

**6. What is the main purpose of a function in programming and scripting?**

- To ensure the program can terminate gracefully.
- To create a reusable chunk of code that performs a specific task.
- To evaluate arithmetic expressions.
- To spawn a new child process in the shell environment.

**7. What is the purpose of a logical operator in programming and scripting?**

- To connect values so that they can be evaluated together.
- To perform addition, subtraction, multiplication, and division.
- To concatenate strings.
- To check whether or not two operands are equal in value.

**8. Which of the following symbols indicates the start of a comment in Bash scripting?**

- ?
- #
- !
- \*

**Activity 13-4: Discussing Writing and Executing a Simple Bash Script****1. You have written a log file for a script that you run to gather output, but each day the output gets overwritten. Which operator can you use to append rather than to overwrite the log file?**

Use the >> append operator to add data to the end of the file each day.

**2. In your script, you want to echo the following message back to the user: "Hello, user. Please select an option to begin." You want "user" to be replaced by the name of the currently logged-in user. How can you do this no matter who is logged in?**

You can use command substitution: echo "Hello, \$(whoami). Please select an option to begin."

- 3. After a storage device controller failure, you restore the repaired system from backups. One of the directories you restore has hundreds of files with extensions such as .ksh, .bsh, and .csh. Unlike Windows, Linux file extensions have nothing to do with what type of file it is. Why might the owner of the files have added these extensions?**

It is true that extensions do not determine the type of file, but it is an easy way to organize files and to know which shell they execute in.

- 4. You are editing an old backup script and find the line source directories.sh in the script. Displaying the contents of the script reveals a list of exported directory names as variables. What is the purpose of using the source command in the backup script?**

The `SOURCE` command executes the other script, `directories.sh`, in the current shell. This provides all the exported variables to the backup script.

- 5. What is the purpose of file globbing in scripts?**

File globbing, such as defining a list of files as `*.sh`, is a shorthand method of defining a list of files with a common name attribute.

## Activity 13-5: Writing and Executing a Simple Bash Script

- 9. From a functionality perspective, how does this script fall short? How could it be improved?**

Answers may vary. The script doesn't easily enable the user to specify a different device to check. The `$part` variable would either need to be explicitly changed, or the script could take in an argument supplied by the user. It would also be helpful if all of the block storage devices on the system were checked, instead of just one. Also, the report itself is very rudimentary and doesn't offer much in the way of guidance. There should be a way to change the output based on the percentage—like a warning for percentages above a certain threshold. A routine that checks for errors would be helpful as well, especially if this script is meant to be called by other scripts.

## Activity 13-6: Discussing Control Statements in Bash Scripts

- 1. What condition does the while loop test for?**

The `while` loop tests for true conditions.

- 2. You are creating a script that tests several conditions. Depending on the test outcome, you want one of three things to happen: send the results to a file, execute another script, or send all output to /dev/null. Which type of conditional statement can you use for this script?**

The `if...elif` statement tests several conditions and can perform different actions based on which condition is met. You can also use the `CASE` statement to achieve the same result.

- 3. You have created a script to check the existence of a file. If the file exists, you want to have the script echo the word "Success" and then exit with an exit code of 0. Otherwise, it should echo "Doesn't exist". Where in the script should you place the `exit 0` code?**

Within the `if...then` branch that tests the condition that the file exists, before the `else` branch.

- 4. True or false? There is a limit to the number of conditions in a `case` statement.**

False. You can have as many conditions as necessary.

- 5. Your colleague has written a script with an array called `names` that holds the names of various users. In the script, a `for` loop iterates through each value in the array. The script is supposed to echo each name to the CLI, but instead, it only echoes the first name in the array multiple times. You confirm that the `for` loop has the proper syntax, so the problem must lie elsewhere. What do you think the problem is with the script?**

It's likely that the `echo` statement within the `for` loop is referencing the `$names` array directly, whereas it should be referencing the iterator variable instead.

## Activity 14-1: Discussing Scheduling Jobs

- 1. You are scheduling new backup tasks on every new Linux system that your team configures and manages. Each requires its key directories to be backed up. Would the `at` command or a cron job be best for scheduling the backup script? Why?**

A cron job is the correct choice because `at` is for running a task once, not for tasks that need repeating.

- 2. Anyone can schedule cron jobs. As a regular user, you need to schedule a file transfer from your local workstation to the shared directory on one of the development servers. You need to create a new cron job. Which command can you use to create the cron job?**

The `crontab -e` command is used to enter the `crontab` editor.

- 3. As the root user, you need to check all scheduled tasks that belong to root. How can you check them?**

Use the `Crontab -l` command to display a list of scheduled tasks.

4.

- How can you remove an entry from the `at` command queue?**

The `atrm` command can be used to delete a scheduled task.

- 5. You have added several jobs to the `at` queue and you need to check the status. How can you check the `at` queue?**

Use the `atq` command.

---

## Activity 14-4: Discussing Version Control Using Git

---

1. You need to set up a new Git repository for a group of developers. Which command begins the process of setting up a new Git repository?

`git init`

2. Your developer group wants to implement Git so that they can manage code versioning. They ask you to install it on the dev01 CentOS server. How can you install Git?

`sudo yum install git`

3. You have created a new file, `statistics.txt`, and want to add it to the Git repository. Which command can you execute to add the file?

`git add statistics.txt`

4. Over the past week, you have made several changes to the `markets.txt` file but have not committed those changes. How can you commit the changes?

Run `git add markets.txt` and then run `git commit -m "Week 1 Changes"`

5. Having returned from vacation today, you want to check the progress of the new coding project your group is working on. Which command can you issue to check the revision history of the repository?

`git log`

---

## Activity 14-6: Identifying Orchestration Concepts

---

1. What is the difference between automation and orchestration? Why does this difference matter to Develetech?

Answers may vary, but automation manages one specific task, while orchestration manages an entire process or combination of individual tasks. It matters because orchestration benefits occur on a much larger scale than automation—for example, with the deployment of an entire web app solution.

2. Give examples of open source orchestration tools used in enterprises.

Ansible, Puppet, Chef, Kubernetes, and OpenStack are all examples of open source orchestration tools.

3. Explain the three general steps in orchestration.

1) Administrators create a set of instructions or configurations. 2) The orchestration tools deliver the instructions and configurations to client systems. 3) The orchestration clients process the instructions and configurations.

4. Ansible is a very popular orchestration tool in both large and small company networks. How does Ansible deliver configuration files to nodes?

Ansible delivers files over SSH connections using an agentless solution.

**5. Develetech will be providing LAMP (Linux, Apache, MySQL, PHP) web hosting packages for customers. How might orchestration benefit Develetech in this context?**

Answers may vary. Orchestration can manage the entire deployment and configuration process of LAMP services. It can automate the deployment steps of Linux, the configuration of the Apache web service, database population of the MySQL database service, and the installation of the PHP programming language (or Perl or Python) components.

**6. Can orchestration help Develetech with cloud-based web hosting? Why or why not?**

Answers may vary, but for the most part, yes, orchestration works well with the cloud solutions of on-demand self-service and rapid elasticity.

---

## Activity 15-1: Preparing to Install Linux

---

**1. Which system would you set up for each purpose, and why?**

Answers may vary. System A should probably go to the graphic designer, who will need lots of RAM, a top-of-the-line video card, and significant storage space for storing images and video files. System B should probably become the web server, as this configuration has the fastest networking capabilities, adequate RAM, fast storage technology, and a processor designed for servers. System C should probably become the test system, because its slower speed is more likely to reflect systems in production throughout the organization and therefore can more clearly illustrate how new software will perform when it is deployed.

**2. As far as the hardware, what else do you need to consider before you can install Linux on these systems?**

Answers will vary, but might include ensuring that System A's video card is supported by the Linux distribution you plan to install; and generally ensuring that all hardware components have adequate driver support in the Linux kernel.

**3. As part of the general roll-out at Develetech, what questions would you need to ask each user to be sure that the new system will meet their needs?**

Answers will vary, but might include: questions that determine what applications users may need to run so you can confirm Linux support as well as adequate hardware requirements; and questions that elicit the network and security requirements for the web server.

**4. Although there are multiple methods of installing Linux onto a system, which is the most common and the likely choice for most users?**

The most common method of installing Linux is booting and installing from removable media, such as DVDs or USB thumb drives.

**5. Assuming you want to use a USB thumb drive to install Linux on these systems, what do you need to consider before doing so?**

Answers may vary, but you need to ensure that the installation files are properly prepared on the drive, rather than just placing the files onto the drive like you would when storing data. You also need to ensure that the BIOS/UEFI environment supports booting from USB.

**6. During installation, why is it important to separate certain directories onto their own file systems?**

Certain directories such as `/home` can fill up and crash the whole Linux server, so it is better to segregate those onto their own file systems, leaving the root file system (`/`) with available space on it.

**7. For Linux servers, which directories should typically be separated onto their own file systems?**

Separate file systems should be created for `/`, `/home`, `/var`, and `swap`.

---

## Activity 15-2: Discussing Installing Linux

---

**1. Depending on the Linux distribution you have chosen to install, which accounts are you prompted to set up during installation?**

Every distribution prompts for a user account and some prompt for the root password as well. CentOS, for example, prompts for a user account and the root password.

**2. During installation, you are prompted to set up a system's networking identity. What information is included in a system's networking identity?**

The hostname, IP address information, and DNS servers. Most networks, especially corporate ones, have DHCP servers that provide the IP address and DNS information.

**3. On larger networks, installing Linux manually each time is not efficient. How do larger organizations install Linux, if not manually?**

They use an automated orchestration suite that distributes a pre-configured system image to the hardware.

**4. You want to manually install Linux on a new laptop that has no DVD drive, and you do not have access to one. How can you install Linux onto the laptop?**

Download the ISO and use a utility to write the ISO to a USB thumb drive. Then, boot from the thumb drive.

**5. What are usually the first two pieces of information you are prompted to provide during a Linux installation?**

You have to select a keyboard layout and system language.



# Glossary

**ABRT**

(Automatic Bug Reporting Tool) A utility that analyzes and reports on problems detected during system runtime.

**absolute path**

A reference to a specific location on a file system irrespective of the current working directory or combined paths.

**ACL**

(access control list) A list of permissions attached to an object.

**agent**

A software program that acts on behalf of some other program or service.

**AppArmor**

A context-based permissions scheme provided with Debian-based and SUSE Linux distributions.

**APT**

(Advanced Package Tool) An advanced package management utility that is used with dpkg packages on Debian-based distributions.

**ARP**

(Address Resolution Protocol) A network protocol that matches MAC addresses to IP addresses.

**array**

A programming object that is a collection of values.

**authentication**

The cybersecurity practice of verifying an individual's identity.

**availability**

The cybersecurity principle of ensuring that computer systems operate continuously and that authorized persons can access the data that they need.

**backup**

A copy of data that exists in another logical or physical location than the original data.

**biometrics**

Authentication schemes that verify a user's identity based on their physical characteristics.

**BIOS**

(Basic Input/Output System) A firmware interface that initializes hardware for an operating system boot.

**blob**

(binary large object) A collection of data that is stored in an unstructured manner.

**block special file**

A type of large file used for data storage.

**Bluetooth**

A short-range wireless radio network transmission medium normally used to connect two personal devices, such as a mobile phone and a wireless headset.

**boot loader**

A small program stored in read-only memory that loads the kernel from a storage device, and then starts the operating system.

**booting**

The process of starting or restarting a computer and loading an operating system for the user to access.

**CA**

(certificate authority) A server that can issue digital certificates and the associated public/private key pairs.

**character special file**

A type of small file used for data streaming.

**chroot jail**

A technique of controlling what a process can access on a file system by changing the root directory of that process's environment.

**CIA triad**

A concept that outlines the three main goals of cybersecurity: confidentiality, integrity, and availability.

**Cinnamon**

One of the default desktop environments for Linux Mint and a fork of GNOME 3.

**CLI**

(command-line interface) A text-based interface between the user and the operating system that accepts input in the form of commands.

**cloud computing**

The practice of delivering on-demand resources to clients over a distributed network like the Internet.

**clustering**

The technique of connecting systems together to perform tasks as if they were one system.

**command mode**

A text editing mode that enables users to perform different editing actions using single keystrokes.

**command substitution**

A method of shell expansion in which the output of a command replaces the command itself.

**compression**

A procedure in which data is encoded to reduce the amount of bits that are used to represent that data.

**conditional statement**

A control statement used in programming to evaluate whether a particular condition is true or false.

**confidentiality**

The cybersecurity principle of keeping information and communications private and protected from unauthorized access.

**console redirection**

The process of forwarding input and output through a serial connection rather than through any I/O peripherals that are directly attached to the system.

**container**

A virtualized application that runs on a host operating system where it is isolated from other containers.

**context-based permissions**

A permissions scheme that describes multiple types of information about processes and files that are used in combination to make decisions related to access control.

**control statement**

A programming element that enables a program to execute instructions in a specified order.

**copyleft**

A type of license that permits users to distribute derivative works of free and open source software (FOSS), as long as the derivative work is released under the same license terms.

**cron job**

A scheduled task that is managed by the Linux CRON daemon.

**CSR**

(certificate signing request) A message sent to a certificate authority (CA) in which an entity applies for a certificate.

**CUPS**

A print management system for Linux that enables a computer to function as a print server.

**CWD**

(current working directory) The directory that the user is currently accessing.

**cybersecurity**

The protection of computer systems and digital information resources from unauthorized access, attack, theft, or data damage.

**daemon**

A program that runs in the background without the need for human intervention.

**decoding**

The process of converting bytes into text.

**desktop environment**

A client to a display server that tells the server how to draw graphical elements on the screen.

**device driver**

A software program that acts as an interface between an operating system and a hardware device.

**device management**

The layer of the Linux kernel that controls device access and interfacing between user applications and hardware.

**device mapping**

The process of abstracting physical storage devices into virtual storage devices.

**DHCP**

(Dynamic Host Configuration Protocol) A network protocol used to automatically assign IP addressing information to hosts.

**differential backup**

A backup type in which all selected files that have changed since the last full backup are backed up.

**digital certificate**

An electronic document that associates credentials with a public key.

**digital signature**

A message digest that has been encrypted with a user's private key.

**display server**

The component of a GUI that constructs and manages visual elements on the screen through a communications protocol.

**distro**

See Linux distribution.

**DM-Multipath**

A feature of the Linux kernel that provides redundancy and improved performance for block storage devices by supporting multiple I/O paths between the CPU and the storage devices.

**DNF**

(Dandified YUM) An improved version of the YUM package manager.

**DNS**

(Domain Name System) A network protocol for translating human-readable hostnames into IP addresses.

**dpkg**

A package management system used by Linux distributions derived from Debian Linux.

**DTLS**

(Datagram Transport Layer Security) A cryptographic network protocol that implements SSL/TLS services using datagrams.

**editing operator**

Keystrokes that can quickly manipulate text in a text editor.

**encoding**

The process of converting text into bytes.

**encryption**

A cryptographic technique that converts data from plaintext form into coded, or ciphertext, form.

**environment variable**

A type of variable whose values are inherited from parent processes and passed on to child processes.

**escape character**

A character that is used to remove the special meaning from another character so it can be interpreted literally.

**execute mode**

A text editing mode that enables users to execute commands within the editor.

**exit code**

A value that a child process passes back to its parent process when the child process terminates.

**exit status**

See exit code.

**ext2**

A file system that used to be the default in older versions of Linux.

**ext3**

A file system that improved upon ext2 in several ways.

**ext4**

One of the default file systems in modern Linux versions that supports journaling and large volumes.

**extended partition**

A storage drive partition that acts as a container for other logical partitions.

**FAT**

(File Allocation Table) An older, simple file system that offers a great deal of compatibility.

**FHS**

(Filesystem Hierarchy Standard) A set of guidelines for the names of files and directories and their locations on Linux systems.

**file system**

A data structure that is used by an operating system to store, retrieve, organize, and manage files and directories on storage devices.

**file system integrity**

The property of a file system that refers to its correctness and validity.

**file system management**

The layer of the Linux kernel that stores, organizes, and tracks files on the system.

**firewall**

A software program or a hardware device that protects a system or a network from unauthorized access by blocking unwanted traffic.

**firewall zone**

The **firewalld** rule sets that can apply to specific network resources, like a network interface.

**FOSS**

(free and open source software) Computer code that embodies the principles of both the free software movement and the open source software (OSS) movement.

**free software**

Computer code that supports the principles set forth in the free software social movement, which focuses on protecting users' rights to view, copy, modify, and distribute software without restriction.

**FSF**

(Free Software Foundation) An organization established by Richard Stallman to promote the free software movement and to support the GNU Project.

**full backup**

A backup type in which all selected files, regardless of prior state, are backed up.

**function**

A block of code that can be reused to perform a specific task.

**GCC**

(GNU Compiler Collection) The standard programming language compiler of the Linux operating system that is used to compile the Linux kernel.

**gedit**

The default GUI text editor used in GNOME desktop environments.

**Git**

A distributed version control system primarily used by developers who are collaborating on projects.

**globbing**

A method of shell expansion used to replace a specific wildcard pattern with values that match the pattern.

**GNOME**

The default desktop environment for most Linux distributions that run the X Window System or Wayland.

**GNU GPL**

(GNU General Public License) A free software license created by the Free Software Foundation (FSF) to support the GNU Project.

**GNU GRUB**

(GNU GRand Unified Bootloader) A boot loader developed by the GNU Project that became popular on Unix-like systems.

**GNU nano**

A small, user-friendly text editor that evolved from the Pico text editor created for Unix-like systems.

**GNU Parted**

A utility that is used to manage partitions on a storage device.

**GNU Project**

The first free software project, one whose goal is to create an operating system that is composed of entirely free software.

**GPIO**

(general-purpose input/output) Pins on a circuit board that have no designated purpose, but are controllable by the user at runtime.

**GPT**

(GUID Partition Table) A modern partition structure that is the successor to the master boot record (MBR).

**group**

An access control object that contains multiple users with similar security requirements.

**GUI**

(graphical user interface) A type of user interface in which users interact with a system or application through visual design elements.

**hard link**

A reference to a file that enables a file's data to have multiple names on the same file system.

**hash**

The theoretically indecipherable fixed-length output of the hashing process.

**hash function**

An algorithm used to calculate the hash value of data.

**hash value**

See hash.

**hashing**

The process of transforming plaintext input into an indecipherable fixed-length output and ensuring that this process cannot be feasibly reversed.

**HBA**

(host bus adapter) A hardware component that connects a host system to a storage device in order to facilitate the input and output of data.

**HCL**

(hardware compatibility list) A database that stores the vendors and models of all hardware devices that an operating system distribution supports in some capacity.

**here document**

A script that takes input from a source and stops processing the input when it reaches a line containing the specified string.

**home directory**

A container for a user's personal files and other files specific to that user.

**hostname**

A human-readable name that identifies a node to a network.

**hotpluggable device**

A hardware device that can be physically added or removed from the system without requiring a reboot in order to use that device.

**HTTP**

(Hypertext Transfer Protocol) A protocol that defines the interaction between a web server and a browser.

**HTTPS**

(Hypertext Transfer Protocol Secure) A version of HTTP that provides a secure connection between a web browser and a server.

**hypervisor**

A layer of software that separates a VM's software from the physical hardware it runs on.

**I/O scheduling**

The process by which the operating system determines the order of input and

output operations as they pertain to block storage devices.

### **IAM**

(identity and access management) A security process that provides identification, authentication, and authorization mechanisms for users, computers, and other entities to work with organizational assets like networks, operating systems, and applications.

### **immutable flag**

An attribute of a file or directory that prevents it from being modified, even by the root user.

### **incremental backup**

A backup type in which all selected files that have changed since the last full or incremental backup (whichever was most recent) are backed up.

### **infrastructure as code**

The process of quickly configuring and deploying infrastructure through programming scripts and other code files, rather than through standard software tools.

### **inheritance**

In file and directory permissions, the property by which an object takes on some of the same permissions that its parent object has.

### **init**

A daemon that initializes a system and acts as the parent to all processes.

### **initrd**

(initial ramdisk) The temporary root file system that is loaded into memory upon system boot.

### **initrd image**

An archive file containing all the essential files that are required for booting the operating system.

### **inode**

(index node) An object that stores metadata about a file or directory on a file system.

### **insert mode**

A text editing mode that enables users to insert text by typing.

### **integrity**

The cybersecurity principle of keeping information accurate, free of errors, and without unauthorized modifications.

### **integrity checking**

The process of verifying that data has not been modified, whether intentionally or unintentionally, in any way.

### **IP address**

(Internet Protocol address) A logical networking address that uniquely identifies a node to a network.

### **IP forwarding**

The Linux kernel implementation of networking routing functionality.

### **IP sets**

Stored collections of IP addresses, network ranges, MAC addresses, port numbers, and network interface names.

### **IPS**

(intrusion prevention system) A security appliance that monitors and evaluates a system for signs of attacks in progress, and can actively block traffic that it determines is malicious.

### **IPSec**

(Internet Protocol Security) A set of open, non-proprietary standards that are used to secure data through authentication and encryption as the data travels across the network or the Internet.

### **ISO image**

An archive file that is often used to package and distribute operating system data for booting and installation.

### **journaling**

A method used by file systems to record changes not yet made to the file system in an object called a journal.

### **KDE Plasma**

The second-most common desktop environment for Linux distributions that run the X Window System or Wayland.

**Kerberos**

A centralized single sign-on (SSO) authentication service that is based on a time-sensitive ticket-granting system.

**kernel**

The core component of an operating system that controls all other components.

**kernel module**

A system-level object that extends the functionality of the kernel.

**kernel panic**

A mechanism by which the system detects there has been a fatal error and responds to it.

**kernel space**

The area of system memory in which the kernel executes the services it provides.

**LDAP**

(Lightweight Directory Access Protocol) A directory service protocol that runs over Transmission Control Protocol/Internet Protocol (TCP/IP) networks.

**library**

In programming, a chunk of compiled code that can be used in programs to accomplish specific common tasks.

**Linux**

A family of operating systems that is based on the Linux kernel and typically incorporates GNU software on top of the kernel.

**Linux distribution**

One of several fully functional operating systems and members of the Linux family that run the Linux kernel, GNU software, and additional components.

**Linux kernel**

A free and open source monolithic kernel that manages all other resources on the Linux operating system.

**load balancing**

The technique of distributing network traffic or computing workload among multiple devices in a network.

**localization**

The process of adapting system components for use within a distinct culture, other than the culture that the system was originally designed for.

**log rotation**

The practice of creating new versions of a log file to maintain a minimum log file size.

**logical partition**

A storage drive partition that is part of an extended partition and can function as a separate drive.

**loop**

A control statement that executes code repeatedly based on a certain condition.

**LUKS**

(Linux Unified Key Setup) A platform-independent full drive/disk encryption (FDE) solution that is commonly used to encrypt storage devices in a Linux environment.

**LVM**

(Logical Volume Manager) Software that maps physical devices and partitions to logical volumes.

**MAC**

(mandatory access control) An access control model in which access is controlled by comparing an object's security designation and a subject's security clearance.

**MAC address**

(media access control address) A unique, physical networking address that is coded into a network interface card (NIC).

**makefile**

A file that contains instructions used by a compiler to build a program from source code.

**manual pages**

Built-in documentation for specific Linux commands that provide information on what a command does and how to use it. Also called man pages.

**MATE**

One of the default desktop environments for Linux Mint and a fork of GNOME 2.

**MBR**

(master boot record) The first physical sector on a storage drive and a type of partition structure.

**MD5**

(Message Digest 5) A deprecated hash function that produces a 128-bit hash value.

**memory management**

The layer of the Linux kernel that allocates memory to user space resources and frees memory when it is no longer in use.

**message digest**

See hash.

**metacharacter**

A special character that a shell interprets in a certain way.

**MFA**

(multi-factor authentication) The practice of requiring a user to present at least two different authentication factors before the system authenticates them.

**microkernel**

A kernel architecture in which the kernel space runs the minimum resources necessary to function as an operating system.

**monolithic kernel**

A kernel architecture in which all system components run in kernel space.

**motion**

A single-key shortcut used to navigate through files in a text editor's command mode.

**mount point**

An access point to information stored on a local or remote storage device.

**name resolution**

The process of translating human-readable domain names into numerical IP addresses.

**Netfilter**

A Linux kernel framework that handles packets that traverse a network interface.

**network adapter**

A device that provides an interface for connecting hosts to a network so that they can exchange data with one another.

**network port**

An endpoint in network communication that identifies the type of application-layer protocol used in a communication.

**NetworkManager**

A Linux utility that is used to configure a system's networking information.

**NFC**

(Near Field Communication) A mobile device communication standard that operates at very short range, often through physical contact.

**NIC**

(network interface card) See network adapter.

**NIC bonding**

The technique of associating two network interfaces together so that they are managed as a single device.

**nice value**

A value that determines the priority by which a process is given CPU time.

**NoSQL**

A type of database design that supports multiple database models beyond the traditional relational database model.

**NTFS**

(New Technology File System) A proprietary file system created by Microsoft as the primary file system for Windows.

**NTP**

(Network Time Protocol) A network protocol that synchronizes a node's time with a designated, definitive time source.

**null device**

A device file that discards all data that is written to it.

**NX**

(NoMachine) Cross-platform proprietary remote desktop software that offers support for multi-session environments.

**octal number**

A number that is expressed using one of eight unique digits, i.e., a base-8 number system.

**off-site backup**

A physical location outside of the main site that stores copies of data.

**OOM killer**

(out-of-memory killer) A feature of the Linux kernel that determines what process(es) to kill when the system is extremely low on memory.

**operator**

A programming object that can evaluate expressions in certain ways.

**orchestration**

The automation of multiple related tasks.

**OSI model**

(Open Systems Interconnection model) A standard that defines how networking is meant to function in terms of different layers.

**OSS**

(open source software) A development methodology in which users are permitted to view, copy, and modify computer code for any reason, as well as distribute it to anyone—all in support of open collaboration.

**OTP**

(one-time password) A password that expires after it is used once, and/or after a small amount of time, to limit its use to a specific individual.

**ownership**

The property by which a user is allowed to apply and modify the permissions of a file or directory.

**package manager**

A program that installs, updates, inventories, and uninstalls packaged software.

**PAM**

(Pluggable Authentication Modules) A framework for centralizing authentication mechanisms leveraged by other services and applications.

**parameter expansion**

See variable substitution.

**parent directory**

The directory that is one level above the current working directory.

**partition**

A section of a storage drive that logically acts as a separate drive.

**path**

A reference to a specific location on a file system.

**PATH variable**

An environment variable that specifies where an application's binaries are stored on the file system.

**PCI**

(Peripheral Component Interconnect) A connection interface standard that is primarily used as an expansion bus for attaching peripheral devices.

**PCIe**

(PCI Express) A connection interface standard that improves upon the original PCI specification in many ways.

**permission string**

The textual representation of an object's permissions.

**permissions**

Access rights assigned to users, enabling them to access and/or modify files and directories.

**PID**

(process ID) A unique integer assigned to each new process when it starts so that it can be identified by the system and users.

**piping**

The process of combining the standard I/O streams of commands.

**PKI**

(public key infrastructure) A system composed of various cryptographic components that authenticate and validate data and entities.

**positional parameter**

A variable within a shell script that is assigned to an argument when the script is invoked.

**primary partition**

A storage drive partition that can contain one file system or logical drive. Sometimes referred to as a volume.

**principle of least privilege**

An information security best practice that states that users should have as much access as is required to do their job, but no more than that.

**privilege escalation**

The act of a user obtaining access to additional resources or functionality that they are normally not allowed access to.

**privileged port**

See trusted port.

**process management**

The layer of the Linux kernel that allocates execution space on the processor for different processes.

**process table**

A record that summarizes the current running processes on a system.

**PXE**

(Preboot Execution Environment) A network boot specification that enables a client to retrieve the necessary boot loader and system files from a server over the network.

**RADIUS**

(Remote Authentication Dial-In User Service) An Internet standard protocol that provides authentication, authorization, and accounting (AAA) services.

**RAID**

(redundant array of independent disks) A set of vendor-independent specifications that support redundancy and fault tolerance for configurations on multiple-device storage systems.

**raw partition**

A partition that enables users and applications to read from and write to a

block storage device directly, without using the system cache.

**redirection**

The process of accepting input data from and sending output data to a component that is not a default I/O device.

**relative path**

A reference to a specific location on a file system that is relative to the current working directory.

**remote desktop**

A concept in which a client connects to a remote system over a network, and is able to sign in to and use the desktop environment of that system.

**repository**

A storage location for software packages.

**router**

A network device that acts as a control point for communications between network segments.

**RPM**

(Red Hat Package Manager) A package management system used by Linux distributions derived from Red Hat Linux.

**runlevel**

In SysVinit, a setting that controls the system's mode of operation.

**SAS**

(Serial Attached SCSI) A computer bus interface and replacement for parallel SCSI that uses serial interface technology.

**SATA**

(Serial AT Attachment) A computer bus interface standard for attaching storage devices to traditional computers.

**SCI**

(System Call Interface) The layer of the Linux kernel that handles system calls sent from user applications to the kernel.

**script**

A computer program that automates the execution of tasks for a particular runtime or shell environment.

**SCSI**

(Small Computer System Interface) A parallel computer bus interface for connecting peripheral devices to traditional computers.

**search path**

A sequence of various directory paths that is used by the shell to locate files.

**sector**

The smallest unit of storage read from or written to a drive.

**SELinux**

(Security-Enhanced Linux) The default context-based permissions scheme provided with CentOS and Red Hat Enterprise Linux.

**service**

Software that responds to requests from other programs to provide some sort of specialized functionality.

**service management**

The lifecycle process of starting services, modifying their running state, and stopping them.

**SGID**

(set group ID) A special permission that enables a user to execute a file with the privileges of the file's group.

**SHA**

(Secure Hash Algorithm) A hash function modeled after MD5 that has multiple versions that produce different sized hash values.

**shell**

A system component that permits users to pass commands and information to the kernel.

**shell environment**

The mechanism by which a shell maintains settings and other behavioral details about the shell.

**shell expansion**

The process by which a shell identifies special tokens that it substitutes values for.

**shell spawning**

The process of creating a new session in a shell environment.

**shell variable**

A type of variable that is contained within specific shell processes and whose values do not get passed on to child processes.

**single-user mode**

A mode of operation in which a Linux system boots into an environment that requires the superuser to log in.

**SNMP**

(Simple Network Management Protocol) A network protocol for collecting and passing device information between hosts.

**software compilation**

The process of translating human-readable source code into computer-readable instructions that the processor can execute.

**SPICE**

(Simple Protocol for Independent Computing Environments) An open source protocol designed specifically for use in a virtual environment.

**SQL**

(Structured Query Language) A programming and query language common to many large-scale database systems.

**SSH**

(Secure Shell) A network protocol that provides an authenticated, encrypted method of connecting to a remote system.

**SSH port forwarding**

The process of tunneling an application through the SSH protocol to encrypt data in transit.

**standard error**

A text stream that is used as the destination for error messages.

**standard input**

A text stream that acts as the source for command input.

**standard output**

A text stream that acts as the destination for command output.

**stderr**

See standard error.

**stdin**

See standard input.

**stdout**

See standard output.

**sticky bit**

A special permission bit that provides protection for files in a directory.

**storage device**

A physical component that can record data and hold it persistently.

**storage quota**

The storage space that is allotted to a user for file storage on a computer.

**strict policy**

In SELinux, a policy in which mandatory access control (MAC) is enforced on all subjects and objects.

**string literal**

A fixed value that represents a string of text within source code.

**SUID**

(set user ID) A special permission that enables a user to execute a file with the privileges of the file's owner.

**superblock**

A data structure that contains a file system's metadata.

**superuser**

The local administrative account on a Linux system, typically named root.

**swap space**

A partition on the storage device that is used when the system runs out of physical memory.

**switch**

A network device that centralizes all network connections for a segment to a single device.

**symbolic link**

A reference to a file or directory that can span multiple file systems.

**syntax**

A set of rules that define the format for issuing commands and writing code in a scripting or programming language.

**syslog**

An event logging standard for Unix-like systems that facilitates centralized logging services over a network.

**system initialization**

The process that begins when the kernel first loads.

**systematic drift**

The predictable amount of time that the hardware clock gains or loses each day, making it inaccurate and throwing it out of alignment with the system clock.

**systemd**

A software suite that replaces older init methods like SysVinit and that includes various service management tools.

**SysVinit**

An older system initialization method that has been largely superseded by systemd.

**tab completion**

A feature in which a command-line shell fills in the name of a command you've partially typed.

**TACACS+**

(Terminal Access Controller Access-Control System Plus) A network authentication protocol that provides authentication, authorization, and accounting (AAA) services with added security features.

**tarball**

An archive file with a .tar extension.

**target**

In systemd, a method of grouping unit files that can be used to set the system's mode of operation.

**targeted policy**

In SELinux, a policy in which mandatory access control (MAC) is enforced on all subjects and objects marked as targeted,

while discretionary access control (DAC) is enforced on all untargeted subjects and objects.

### **TCP/IP**

(Transmission Control Protocol/Internet Protocol) The networking protocol that governs Internet communications and many other modern computer networks.

### **text editor**

An application that enables you to view, create, or modify the contents of text files.

### **text stream**

A sequence of one or more lines of text that applications can leverage to read from or write to a particular device or system component.

### **thin client**

Any lightweight computing device that connects to a more powerful server for doing work.

### **token**

A unique object, whether physical or digital, that a user possesses and that can be used to verify that user's identity.

### **troubleshooting**

The process of recognizing, diagnosing, and resolving problems.

### **troubleshooting model**

A step-by-step approach to the troubleshooting process.

### **trusted port**

A network port in the well-known range (0–1023) that requires superuser privileges for any services attempting to use this port.

### **udev**

A device manager that automatically detects and configures hardware devices.

### **UEFI**

(Unified Extensible Firmware Interface) A firmware interface that improves upon and is meant to succeed BIOS firmware.

### **UFW**

(Uncomplicated Firewall) A simplified interface for configuring the **iptables** firewall service.

### **unit file**

A configuration file that systemd uses to determine how it will handle system resources that systemd can manage.

### **Unix philosophy**

A set of best practices and approaches to software development, proposed by the lead developers of the Unix operating system, that emphasize simplicity and modularity.

### **USB**

(Universal Serial Bus) The *de facto* standard interface technology for connecting peripheral devices to computers.

### **user account**

An object that represents a person's identity to the operating system and is used to perform certain tasks.

### **user space**

The area in system memory outside of kernel space that includes software that accesses kernel services.

### **variable**

An entity whose value changes from time to time.

### **variable assignment**

The act of defining a variable as having a certain value.

### **variable substitution**

The act of referencing or retrieving the value of a variable.

### **VFS**

(virtual file system) An abstraction layer that translates file system information between a real file system and the Linux kernel.

### **Vim**

An extended version of the vi text editor.

### **virtualization**

The process of creating a simulation of a computing environment, where the virtualized system can simulate the hardware, operating system, and applications of a typical computer without being a separate physical computer.

**visual mode**

A text editing mode that enables users to highlight or select text for copying, deleting, and so on.

**VM**

(virtual machine) A virtualized computer that consists of an operating system and applications that run in a virtual environment that simulates dedicated physical hardware.

**VNC**

(Virtual Network Computing) A cross-platform remote desktop service that uses the Remote Frame Buffer (RFB) protocol.

**VPN**

(virtual private network) A method of extending a private network by tunneling through a public network, such as the Internet.

**Wayland**

A display server and reference implementation in Unix-like operating systems that is meant to improve upon and replace the X Window System.

**Wi-Fi**

A wireless communications protocol for establishing and connecting to a wireless local area network (WLAN).

**window manager**

See desktop environment.

**Wireshark**

A common network packet sniffer and analysis tool.

**X**

See X Window System.

**X Window System**

A platform-independent display server and windowing system that was developed by the Massachusetts Institute of Technology (MIT) in 1984.

**X.Org Server**

A free and open source reference implementation of the X Window System for Linux and other Unix-like operating systems.

**X11**

See X Window System.

**XFS**

One of the default file systems in modern Linux versions that supports journaling and efficiency in handling large files.

**xrdp**

An open source utility that creates a Remote Desktop Protocol (RDP)-like environment for non-Windows systems.

**YUM**

(Yellowdog Updater, Modified) An advanced package management utility that is used with RPM packages on Red Hat-based distributions.

**Zypper**

An openSUSE package manager that supports .rpm packages.

# Index

## A

ABRT [374](#)

absolute paths [157](#)

Access Control Lists, *See* ACLs ACLs [96](#)

Address Resolution Protocol, *See* ARP

Advanced Package Tool, *See* APT agents [545](#)

AppArmor [523](#)

APT [456](#)

ARP [444](#)

arrays [590](#)

authentication [493](#)

Automatic Bug Reporting Tool, *See* ABRT

availability [493](#)

## B

backups

off-site [558](#)

Bash

scripting [588](#)

Basic Input/Output System, *See* BIOS

binary large object, *See* blob

biometrics [494](#)

BIOS [263](#)

blob [425](#)

block special files [154](#)

Bluetooth [347](#)

bonding [407](#)

booting [262](#)

boot loaders [262](#)

## C

CA [391, 513](#)

certificate authority, *See* CA certificate

signing request, *See* CSR character

special files [154](#)

chroot jail [496](#)

CIA triad [492](#)

CIFS [116](#)

Cinnamon [299](#)

## CLI

advantages and challenges [12](#)

cloud computing [420](#)

cluster [393](#)

command-line interface, *See* CLI

command mode [179](#)

command substitution [598](#)

comments [592](#)

Common Internet File System, *See* CIFS

compile [454](#)

compilers [481](#)

compression [563](#)

conditional statements [605](#)

confidentiality [492](#)

console redirection [301](#)

containers [395](#)

context-based permissions [520](#)

control statements [605](#)

copyleft [4](#)

cron jobs [621](#)

CSR [513](#)

CUPS [357](#)

current working directory, *See* CWD CWD

[157](#)

cybersecurity [492](#)

## D

daemons [308](#)

Dandified YUM, *See* DNF

Datagram Transport Layer Security, *See* DTLS

decoding [289](#)

desktop environment [298](#) *See*

*also* window manager

device drivers [239](#)

device management [241](#)

device mapping [130](#)

DHCP [384, 392, 412](#)

differential backups [553](#)

digital certificates [513](#)

digital signatures [513](#)

display servers [296](#)

- distros [6](#)  
 DM-Multipath [130](#)  
 DNF [456](#)  
 DNS [384](#), [391](#)  
 Domain Name System, See DNS  
 dpkg [454](#)  
 DTLS [515](#)  
 Dynamic Host Configuration Protocol, See DHCP
- E**
- editing operators [181](#)  
 encoding [289](#)  
 encryption [497](#)  
 environment variables [579](#)  
 escape character [590](#)  
 execute mode [179](#)  
 exit code [596](#)  
*See also* exit status  
 exit status [596](#)  
 ext2 [115](#)  
 ext3 [115](#)  
 ext4 [115](#)
- F**
- FAT [115](#)  
 FHS [155](#)  
 File Allocation Table, See FAT  
 Filesystem Hierarchy Standard, See FHS  
 file system integrity [146](#)  
 file system management [241](#)  
 file systems  
     types of [115](#)  
 firewalls [529](#)  
 FOSS [3](#)  
 free and open source software, See FOSS  
 free software [3](#)  
 Free Software Foundation, See FSF  
 FSF [4](#)  
 full backups [553](#)  
 functions [591](#)
- G**
- GCC [253](#)  
 gedit text editor [183](#)  
 general-purpose input/output, See GPIO
- Git [627](#)  
 globbing [598](#)  
 GNOME [299](#)  
 GNU Compiler Collection, See GCC GNU  
 General Public License, See GPL GNU  
 GRand Unified Bootloader, See GNU  
 GRUB  
 GNU GRUB [274](#)  
 GNU nano [182](#)  
 GNU Parted  
     menu options [121](#)  
 GNU Project [3](#)  
 GPIO [349](#)  
 GPL [4](#)  
 GPT [265](#)  
 graphical user interface, See GUI groups  
[56](#)  
 GUI [295](#)  
 GUID Partition Table, See GPT
- H**
- hard links [218](#)  
 hardware compatibility list, See HCL hash functions [569](#)  
 hashing [499](#)  
 HBA [350](#)  
 HCL [641](#)  
 home directory [156](#)  
 host bus adapter, See HBA  
 hostname [383](#)  
 hotpluggable devices [354](#)  
 HTTP [391](#)  
 HTTPS [391](#)  
 Hypertext Transfer Protocol, See HTTP  
 Hypertext Transfer Protocol Secure, See HTTPS  
 hypervisors [422](#)
- I**
- I/O scheduling [164](#)  
 IAM [507](#)  
 identity and access management, See IAM immutable flag [95](#)  
 incremental backups [553](#)  
 index nodes, See inodes  
 infrastructure as code [634](#) init [308](#)  
 initial ramdisk, See initrd

initrd  
 image 266  
 inodes 116  
 insert mode 179  
 integrity 493  
 integrity checking 569  
 Internet Protocol Security, See IPSec  
 intrusion prevention systems, See IPSs IP address 383  
 IP forwarding 534  
 IPSec 514  
 IP sets 535  
 IPSs 536  
 ISO image 264

**J**

journaling 117

**K**

KDE Plasma 299  
 Kerberos 495  
 kernel modules 245  
 kernel panic 269  
 kernels  
   microkernel 239  
   monolithic 239  
 kernel space 238

**L**

LDAP 495  
 libraries 481  
 Lightweight Directory Access Protocol,  
*See LDAP*  
 Linux  
   advantages of 5  
   disadvantages of 5  
   documentation 24  
   list of distributions 6  
   operating system family 4  
   uses for 7  
 Linux distributions 6  
*See also distros*  
 Linux kernel 239  
 Linux Unified Key Setup, See LUKS load balancing 393  
 localization 284  
 Logical Volume Manager, See LVM  
 log rotation 544  
 loops 609

LUKS 498  
 LVM  
   advantages 131  
   tools 131

**M**

MAC 520  
 MAC address 383  
 makefile 482  
 mandatory access control, See MAC  
 man pages 24  
 manual pages 24  
   *See also* man pages master boot record, See MBR MATE 300  
 MBR 265  
 MD5 569  
   memory management 241  
   Message Digest 5, See MD5  
   metacharacters 595  
   MFA 495  
   microkernels 239  
   monolithic kernels 239  
   motions 181  
   mount points 138  
   multi-factor authentication, See MFA

**N**

Near Field Communication, See NFC  
 Netfilter 533  
 network adapters 349  
 Network File System, See NFS  
 network interface cards, See NICs  
 NetworkManager 398  
 network ports 386  
 Network Time Protocol, See NTP  
 New Technology File System, See NTFS  
 NFC 347  
 NFS 116  
 nice value 325  
 NICs 349  
 NoMachine, See NX NoSQL 394  
 NTFS 115  
 NTP 390  
 null device 230  
 NX 301

**O**

off-site backups [558](#)  
 one-time passwords, *See* OTPs OOM killer [339](#)  
 open source software, *See* OSS  
 Open Systems Interconnection model, *See* OSI model operators [589](#)  
 orchestration [634](#)  
 OSI model [382](#)  
 OSS [2](#)  
 OTPs [494](#)  
 out-of-memory killer, *See* OOM killer ownership [87](#)

**P**

package managers [454](#)  
 PAMs [510](#)  
 parameter expansion [589](#)  
 parent directory [157](#)  
 partitions  
   extended [118](#)  
   logical [118](#)  
   primary [118](#)  
 paths [157](#)  
 PATH variable [192](#)  
 PCI [350](#)  
 PCIe [350](#)  
 PCI Express, *See* PCIe  
 Peripheral Component Interconnect, *See* PCI  
 permissions  
   attributes [77](#)  
   contexts [77](#)  
   default [82](#)  
   special [93](#)  
 permission string [77](#)  
 PIDs [321](#)  
 pipes [228](#)  
 piping [228](#)  
 PKI  
   components [513](#)  
 Pluggable Authentication Modules, *See* PAMs  
 positional parameters [598](#)  
 Preboot Execution Environment, *See* PXE  
 principle of least privilege [37](#)  
 privileged ports [536](#)  
 privilege escalation [496](#)

process IDs, *See* PIDs process

management [241](#)  
 process table [322](#)  
 profiles [68](#)  
 public key infrastructure, *See* PKI PXE [264](#)

**Q**

quota management [168](#)  
 quota reports [168](#)

**R**

RADIUS [494](#)  
 RAID [130](#)  
 raw partition [265](#)  
 Red Hat Package Manager, *See* RPM  
 redirection  
   operators [226](#)  
 redundant array of independent disks, *See* RAID  
 relative paths [157](#)  
 Remote Authentication Dial-In User Service, *See* RADIUS  
 remote desktop [300](#)  
 repositories [470](#)  
 router [384](#)  
 RPM [454](#)  
 runlevels [313](#)

**S**

SAS [350](#)  
 SATA [350](#)  
 schedulers  
   types of [164](#)  
 SCI [241](#)  
 scripts [578](#)  
 SCSI [350](#)  
 search paths [582](#)  
 sectors [264](#)  
 Secure Hash Algorithm, *See* SHA  
 Secure Shell, *See* SSH  
 Security-Enhanced Linux, *See* SELinux  
 SELinux [520](#)  
 Serial AT Attachment, *See* SATA  
 Serial Attached SCSI, *See* SAS Server Message Block, *See* SMB service management [308](#)  
 services [308](#)  
 setgid [93](#)

- set group ID, *See* setgid  
*See also* SGID
- setuid 93
- set user ID, *See* setuid  
*See also* SUID
- SGID 93
- SHA 570
- shell environment 578
- shell expansion 597
- shells 12
- shell spawning 578
- shell variables 579
- Simple Network Management Protocol, *See* SNMP
- Simple Protocol for Independent Computing Environments, *See* SPICE
- single-user mode 313
- Small Computer System Interface, *See* SCSI
- SMB 116
- SNMP 392
- software compilation 482
- SPICE 301
- SQL 394
- SSH
- authentication 507
  - port forwarding 302
  - services 390
- standard error 226 *See also* stderr
- standard input 226 *See also* stdin
- standard output 226 *See also* stdout
- stderr 226
- stdin 226
- stdout 226
- sticky bits 94
- storage devices 114
- storage quotas 167
- string literals 590
- Structure Query Language, *See* SQL
- substitution 589
- See also* parameter expansion
- SUID 93
- superblock 148
- superuser 37
- swap space 119
- switch 384
- switch modes 179
- symbolic links 219
- syntax 588
- syslog 543
- systematic drift 288
- System Call Interface, *See* SCI systemd 309
- system initialization 308
- SysVinit 312

**T**

- tab completion 17
- TACACS+ 494
- tarball 477
- targets 310
- TCP/IP 382
- templates 423
- Terminal Access Controller Access-Control System Plus, *See* TACACS+
- text editors 178
- text streams 226
- thin clients 346
- tokens 494
- Transmission Control Protocol/Internet Protocol, *See* TCP/IP
- troubleshooting
- models 105
  - strategies 105
- trusted ports 536
- See also* privileged ports

**U**

- udev 355
- UEFI 263
- UFW 531
- Uncomplicated Firewall, *See* UFW
- Unified Extensible Firmware Interface, *See* UEFI
- unit files 309
- Universal Serial Bus, *See* USB
- Unix philosophy 4
- USB 347
- user accounts
- types of 36
- user space 238

**V**

- variable assignment 588
- variables
- environment 579
  - shell 579

variable substitution [589](#)  
VFS [117](#)  
Vi IMproved, *See* Vim  
Vim [178](#)  
Vim modes [179](#)  
virtual file system, *See* VFS  
virtualization [394](#)  
virtual machines, *See* VMs  
Virtual Network Computing, *See* VNC  
virtual private network, *See* VPN visual mode [179](#)  
VMs [394](#)  
VNC [300](#)  
VPN [394](#)

## W

Wayland [297](#)  
Wi-Fi [347](#)  
window manager [298](#)  
Wireshark [439](#)

## X

X.Org Server [296](#)  
X11 [296](#)  
XFS  
    tools [149](#)  
xrdp [300](#)  
X Window System [296](#)  
    *See also* X11

## Y

Yellowdog Updater, Modified, *See* YUM  
YUM [455](#)

## Z

Zypper [456](#)



ISBN-13 978-1-6427-4153-7  
ISBN-10 1-6427-4153-1



9 781642 741537 90000