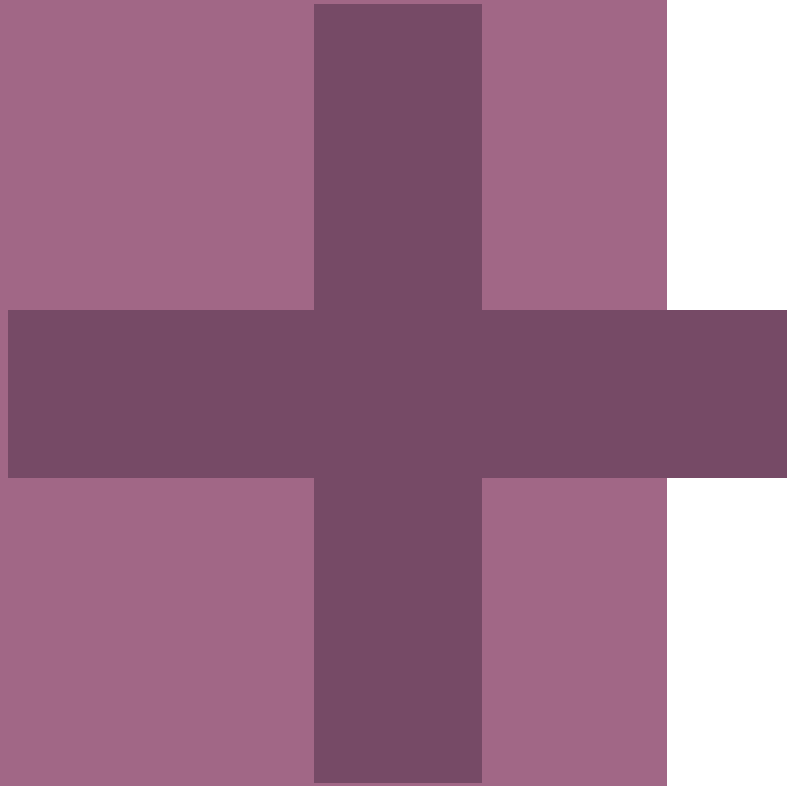


CompTIA®



The Official CompTIA

Linux+

Study Guide

Exam XK0-005



Official CompTIA Content Series for CompTIA Performance Certifications

**The Official
CompTIA
Linux+
Study Guide
(Exam XK0-005)**

Acknowledgments



Damon Garn, Author

Becky Mann, Director, Product Development

James Chesterfield, Senior Manager, User Experience and Design

Katherine Keyes, Senior Specialist, Product Development

Notices

Disclaimer

While CompTIA, Inc. takes care to ensure the accuracy and quality of these materials, we cannot guarantee their accuracy, and all materials are provided without any warranty whatsoever, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. The use of screenshots, photographs of another entity's products, or another entity's product name or service in this book is for editorial purposes only. No such use should be construed to imply sponsorship or endorsement of the book by nor any affiliation of such entity with CompTIA. This courseware may contain links to sites on the Internet that are owned and operated by third parties (the "External Sites"). CompTIA is not responsible for the availability of, or the content located on or through, any External Site. Please contact CompTIA if you have any concerns regarding such links or External Sites.

Trademark Notice

CompTIA®, Linux+®, and the CompTIA logo are registered trademarks of CompTIA, Inc., in the U.S. and other countries. All other product and service names used may be common law or registered trademarks of their respective proprietors.

Copyright Notice

Copyright © 2022 CompTIA, Inc. All rights reserved. Screenshots used for illustrative purposes are the property of the software proprietor. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of CompTIA, 3500 Lacey Road, Suite 100, Downers Grove, IL 60515-5439.

This book conveys no rights in the software or other products about which it was written; all use or licensing of such software or other products is the responsibility of the user according to terms and conditions of the owner. If you believe that this book, related materials, or any other CompTIA materials are being reproduced or transmitted without permission, please call 1-866-835-8020 or visit <https://help.comptia.org>.

Table of Contents

Lesson 1: Introducing Linux	1
Topic 1A: Identify Linux Characteristics.....	2
Topic 1B: Understand Bash Interaction with Linux.....	9
Topic 1C: Use Help in Linux.....	20
Topic 1D: Identify the Linux Troubleshooting Methodology.....	24
Lesson 2: Administering Users and Groups	31
Topic 2A: Manage User Accounts.....	32
Topic 2B: Manage Group Accounts.....	43
Topic 2C: Configure Privilege Escalation.....	47
Topic 2D: Troubleshoot User and Group Issues.....	54
Lesson 3: Configuring Permissions	61
Topic 3A: Configure Standard Linux Permissions.....	62
Topic 3B: Configure Special Linux Permissions.....	74
Topic 3C: Configure Access Control Lists.....	77
Lesson 4: Implementing File Management	83
Topic 4A: Understand the Linux File System.....	84
Topic 4B: Use File Management Commands.....	92
Topic 4C: Find File Locations.....	108
Lesson 5: Authoring Text Files	119
Topic 5A: Edit Text Files.....	120
Topic 5B: Manage Text Files.....	128
Lesson 6: Managing Software	139
Topic 6A: Understand Software Management.....	140
Topic 6B: Manage RPM Software Packages and Repositories.....	145
Topic 6C: Manage Debian-based Software Packages and Repositories.....	157

- Topic 6D: Compile from Source Code..... 162
- Topic 6E: Acquire Software 165
- Topic 6F: Run Software in a Sandbox..... 168

- Lesson 7: Administering Storage..... 173**
 - Topic 7A: Understand Storage 174
 - Topic 7B: Deploy Storage..... 179
 - Topic 7C: Manage Other Storage Options 197
 - Topic 7D: Troubleshoot Storage 205

- Lesson 8: Managing Devices, Processes, Memory, and the Kernel..... 215**
 - Topic 8A: Gather Hardware Information 216
 - Topic 8B: Manage Processes 223
 - Topic 8C: Manage Memory..... 234
 - Topic 8D: Manage the Linux Kernel 239

- Lesson 9: Managing Services 251**
 - Topic 9A: Manage System Services 252
 - Topic 9B: Configure Common System Services..... 263
 - Topic 9C: Configure Localization Settings 277

- Lesson 10: Configuring Network Settings 285**
 - Topic 10A: Understand Network Fundamentals 286
 - Topic 10B: Manage Network Settings..... 292
 - Topic 10C: Configure Remote Administrative Access 305
 - Topic 10D: Troubleshoot the Network 313

- Lesson 11: Configuring Network Security..... 327**
 - Topic 11A: Configure the Firewall 328
 - Topic 11B: Monitor Network Traffic 337

Lesson 12: Managing Linux Security	355
Topic 12A: Harden a Linux System.....	356
Topic 12B: Manage Certificates	363
Topic 12C: Understand Authentication	369
Topic 12D: Configure SELinux or AppArmor.....	375
Lesson 13: Implementing Simple Scripts	385
Topic 13A: Understand Bash Scripting Basics.....	386
Topic 13B: Use Shell Script Elements	390
Topic 13C: Implement Scripts with Logical Controls.....	401
Lesson 14: Using Infrastructure as Code	419
Topic 14A: Understand Infrastructure as Code	420
Topic 14B: Implement Orchestration	424
Topic 14C: Manage Version Control with Git	429
Lesson 15: Managing Containers in Linux	437
Topic 15A: Understand Containers	438
Topic 15B: Deploy Containers	441
Topic 15C: Understand Virtualization Concepts.....	448
Lesson 16: Installing Linux	453
Topic 16A: The Linux Boot Process.....	454
Topic 16B: Modify Boot Settings.....	462
Topic 16C: Deploy Linux.....	468
Appendix A: Mapping Course Content to CompTIA Linux+ (Exam XK0-005)	A-1
Appendix B: Linux Command Reference Guide	B-1
Solutions	S-1
Glossary	G-1
Index	I-1

About This Course

CompTIA is a not-for-profit trade association with the purpose of advancing the interests of IT professionals and IT channel organizations; its industry-leading IT certifications are an important part of that mission. CompTIA's Linux+ certification is an intermediate-level certification designed for professionals with at least 12 months of hands-on experience working with Linux servers in a junior Linux support engineer or junior cloud/DevOps support engineer job role. In addition, the knowledge gained in CompTIA's A+, Network+, and Server+ courses, or the equivalent, is strongly recommended.

The CompTIA Linux+ certification exam will verify the successful candidate has the knowledge and skills required to configure, manage, operate, and troubleshoot Linux in on-premises and cloud-based server environments while using security best practices, scripting, containerization, and automation.

CompTIA Linux+ Exam Objectives

Course Description

Course Objectives

This course can benefit you in two ways. If you intend to pass the CompTIA Linux+ (Exam XK0-005) certification examination, this course can be a significant part of your preparation. But certification is not the only key to professional success in the field of systems administration. Today's job market demands individuals with demonstrable skills, and the information and activities in this course can help you build your sysadmin skill set so that you can confidently perform your duties in any intermediate-level Linux systems administration role.

On course completion, you will be able to:

- Configure, manage, and troubleshoot Linux systems.
- Operate Linux in both on-premises and cloud-based server environments.
- Implement security best practices.
- Use scripting, containerization, and automation to optimize a Linux system.

Target Student

The Official CompTIA Linux+ (Exam XK0-005) is the primary course you will need to take if your job responsibilities include Linux system administration, installation, and security within your organization. You can take this course to prepare for the CompTIA Linux+ (Exam XK0-005) certification examination.

Prerequisites

To ensure your success in this course, you should have at least 12 months of hands-on experience working with Linux servers. CompTIA A+, Network+, and Server+ certifications, or the equivalent knowledge, are strongly recommended.



The prerequisites for this course 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: www.comptia.org/training/resources/exam-objectives.

How to Use the Study Notes

The following notes will help you understand how the course structure and components are designed to support mastery of the competencies and tasks associated with the target job roles and will help you prepare to take the certification exam.

As You Learn



At the top level, this course is divided into **Lessons**, with each representing an area of competency within the target job roles. Each Lesson is composed of a number of topics. A **Topic** contains subjects that are related to a discrete job task and mapped to objectives and content examples in the CompTIA exam objectives document. Rather than follow the exam domains and objectives sequence, lessons and topics are arranged in order of increasing proficiency. Each topic is intended to be studied within a short period (typically 30 minutes at most). Each topic is concluded by one or more activities, designed to help you apply your understanding of the study notes to practical scenarios and tasks.

In addition to the study content in the lessons, there is a glossary of the terms and concepts used throughout the course. There is also an index to assist in locating particular terminology, concepts, technologies, and tasks within the Lesson and topic content.



In many electronic versions of the book, you can click links on key words in the topic 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 eBook viewing software.

Watch throughout the material for the following visual cues.

Student Icon	Student Icon Descriptive Text
	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.

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.

Following the lesson content, you will find a table mapping the lessons and topics to the exam domains, objectives, and content examples. You can use this as a checklist as you prepare to take the exam and review any content that you are uncertain about.

As a Reference

The organization and layout of this book make it an easy-to-use resource for future reference. Guidelines can be used during class and as after-class references when you're back on the job and need to refresh your understanding. When 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.

Lesson 1

Introducing Linux

LESSON INTRODUCTION

Working with Linux begins with an understanding of licensing and the operating system's history. The open-source nature of Linux has resulted in many different distributions, so it's important to understand how distributions differ from each other. Linux servers are primarily managed from the command line, using shells such as Bash. Bash enforces a particular syntax, or way of structuring commands. In addition, Linux holds its configurations in text files, so it's critical that sysadmins can edit these files to manage system settings. Man pages are available as quick reference documents to help administrators recall the function of specific commands and any available options.

Misconfigurations or physical failures may provide troubleshooting opportunities, so sysadmins should follow a standard methodology to help narrow the scope of problems, solve the root cause of the issue, and manage documentation related to configuration issues.

Lesson Objectives

In this Lesson, you will:

- Identify Linux characteristics.
- Understand basic interaction with Linux.
- Use help documentation in Linux.
- Identify the troubleshooting methodology.

Topic 1A

Identify Linux Characteristics



EXAM OBJECTIVES COVERED

This topic provides background information about the history and features of Linux and does not cover a specific exam objective.

Linux is characterized by a free and open-source software licensing approach, which makes the operating system freely distributable and encourages the release of modified versions. These versions are known as distros or distributions. Distros are purpose-specific combinations of the Linux OS and particular applications geared toward supporting defined goals, such as enterprise services, database management, or virtualization hosting. Finally, Linux is distinctive for frequently being managed from a command-line interface, which is often more efficient and easier on resources, rather than a graphical interface.

Characteristics of Free and Open-Source Software

Much of today's software is closed-source software, or proprietary software that's released under copyright law. These laws restrict intellectual property, such as closed-source software, from duplication and re-release by competitors. **Open-source** software takes a different approach.

In general, software licensed as open-source can be duplicated, shared, and modified, and the modified versions can be released to consumers. Code licensed as **free and open-source software (FOSS)** can be used and changed without cost. In fact, changes are encouraged as a form of improvement, even by individuals who do not work for the original developers. Any changes must also be made available and released for free.



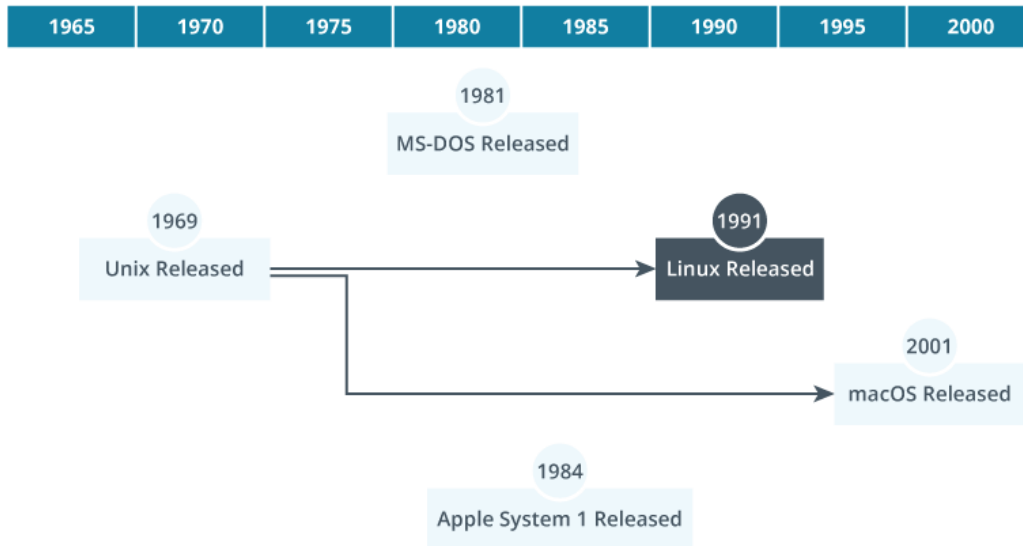
Some examples of open-source licenses include the [Apache License 2.0](#), the [GNU General Public License](#), and [Mozilla's Public License](#). While the exam does not focus on the specifics of these licenses, they are a good sample of the open-source requirements and permissions.

The History and Philosophy of Unix and Linux

Unix is one of the oldest operating systems still in use. It was created in 1969, and it was not released as open-source software. Instead, Unix versions were associated with many different tech organizations, including IBM, Hewlett-Packard, and AT&T. These various Unix versions are referred to as Unix "flavors" and were proprietary to each company.

In 1991, Linus Torvalds created a new Unix-like operating system kernel. He released this kernel, which he called **Linux**, under the GPL license. The **Linux kernel**, as well as much of the software released with it, is open-source; it can be modified, shared freely, and re-released. This collaborative approach allows Linux to grow and evolve rapidly. As a result of this approach, there are now more than 200 Linux versions, or distributions (abbreviated "distros").

Of the three primary operating systems in the marketplace today (Linux, macOS, and Windows), two can trace their roots back to Unix. The macOS kernel evolved from a Unix flavor named BSD and shares many of the same standards and some software as Linux. However, Apple's OS is not FOSS. Microsoft Windows also uses a proprietary kernel with a more restrictive licensing method.



The timeline of early OS development. Unix, released in 1969, directly generated the Linux and the macOS systems.

Traits of the Linux Operating System

Like any operating system, Linux has characteristics that may or may not fit the needs of a given organization. Here are a few general considerations:

- **Free:** No licensing fees or tracking associated with most Linux distributions.
- **Security:** Because of the open-source nature of Linux and its associated software, many developers can and do review code for vulnerabilities. Such vulnerabilities tend to be addressed quickly.
- **Support:** Community-driven support may provide easy, efficient, and cost-effective solutions. However, support may be limited to the community, without a strong corporate support structure implemented by the distribution's vendor.
- **Performance:** Linux often provides greater performance and stability compared to other operating systems.
- **Software availability:** Fewer or less familiar software options may exist, especially for nonbusiness applications, such as games.
- **Hardware requirements:** Linux may consume fewer hardware resources, making it easier to retain older systems for longer.
- **Hardware flexibility:** Linux runs on a wide variety of hardware platforms, adding to its flexibility in areas such as Internet of Things (IoT). Specialized hardware may require specific drivers that may not exist for Linux.
- **Learning curve:** Some find that Linux has a steeper learning curve than Windows or macOS does.

- **Distribution creation:** If existing Linux distributions do not fit your needs, you are welcome (and encouraged) to create your own. The sheer number and purpose of Linux distributions can be confusing and overwhelming. There is not a big name in the marketplace that represents Linux and lends it a sense of stability.

Understand Linux Distributions

Because anyone can create and release their own version of Linux, there are thousands of different options. These individual releases are called distributions (or “**distros**” for short). Distributions are purpose-specific versions of Linux that address a specific need, such as system security or application hosting.

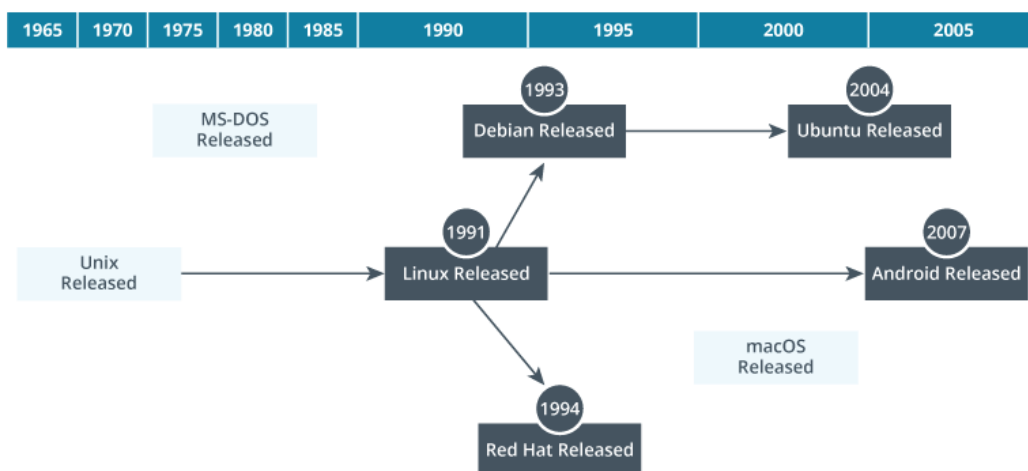
Many distributions trace their history back to one of two specific Linux distributions: Red Hat Linux or Debian Linux. One of the main differentiators between these two distros is how they manage software. Those distros derived from Red Hat Linux use different software managers than those derived from Debian Linux. The software is also packaged differently.



Software management, including more differences between the Red Hat method and the Debian method, are discussed in a later section.

Some of the most common distros include:

- Fedora Linux
- Ubuntu Desktop, Server, Core
- Red Hat Enterprise Linux (RHEL)
- Linux Mint
- Debian
- openSUSE



After the release of Linux in 1991, the two major branches, Debian and Red Hat, followed quickly and generated hundreds of distros.

Many of these distributions fulfill specific roles in the marketplace, including desktop or workstation computer, server, IoT device, mobile device, or other functions. While mobile and IoT implementations are common, the focus of this course is on server deployments. One of the most important characteristics of a distribution is its included software.

Some distributions contain end-user applications, such as word processors or presentation software. Others contain server services, such as web services or file storage. Still other distributions include security software or creative applications, such as music editing.

Linux server deployments are put to use in the following ways:

- **Webserver:** Hosts one or more websites.
- **Name resolution:** Hosts Domain Name System (DNS) name resolution services.
- **File:** Stores business data, usually in some form of text document.
- **Print:** Manages the print process and access to print services.
- **Log:** Centralizes and stores log files from other systems.
- **Virtualization/container:** Hosts virtual machine or container virtualization software.
- **Database:** Hosts one or more databases.
- **Cluster:** Works with other cluster nodes to host high-performance, fault-tolerant services.

Linux is heavily involved in newer forms of infrastructure management. A DevOps approach to the management of such Linux servers and services works toward high quality, iterative, and frequent updates and releases. Linux tends to include security in design and implementation throughout the development lifecycle (this approach is sometimes called DevSecOps).



Most commands are consistent across distributions. A few commands, such as those for software management, may be specific to one group of distributions or another. For example, Red Hat Linux uses the `rpm` command to manage software, while Debian Linux uses `apt`.

The Command-Line Interface

One distinguishing characteristic of Linux compared to other operating systems is its reliance on the **command-line interface (CLI)**. Linux administrators frequently use the CLI for everyday tasks, while administrators of other platforms often use **graphical user interface (GUI)** utilities. In fact, the installation of a GUI is often optional with Linux and may be frowned upon for performance and security reasons.

A GUI consumes a great many hardware resources, specifically memory and processor time. On a server, these resources should be dedicated to the service provided, such as handling database queries or managing print jobs. Desktop systems might need a user-friendly GUI but servers usually do not.

CLI advantages:

- **Quicker:** It's usually quicker to execute a series of commands at the CLI (assuming you know the commands).

- **Performance:** CLI environments consume fewer hardware resources, leaving those resources free to support the server's purpose.
- **Scriptable:** CLI commands can be written into a text file, which the system then reads and executes in a consistent, efficient, repeatable, and scheduled manner.

CLI disadvantages:

- **Learning curve:** Remembering many different commands and their related options is difficult.
- **Nonintuitive:** Commands are often difficult to relate to or understand, with no apparent logic.
- **Inconsistent:** Many commands differ from each other in small but distinctive ways, making it difficult to recall exactly how to use them.

Common CLIs

Command-line interfaces are available in Linux, Windows, and macOS. Users type commands using a specific syntax, and the system processes the commands. At first, such input may seem intimidating or difficult, but CLI environments get easier with use. These environments are usually faster and offer automation options that are not available in GUIs.

```
student@ubuntu20:~$ whoami
student
student@ubuntu20:~$ pwd
/home/student
student@ubuntu20:~$ date
Fri 12 Nov 2021 11:36:22 AM MST
student@ubuntu20:~$ █
```

Several sample commands and their output, including `whoami`, `pwd`, and `date`.

Shells provide the CLI. Each shell has its own syntax, or way of structuring commands.

Common Linux shells:

- **Bash:** Default Linux shell
- **ksh:** Korn shell
- **zsh:** Z shell

These shells are differentiated by their syntax and user-friendly features.



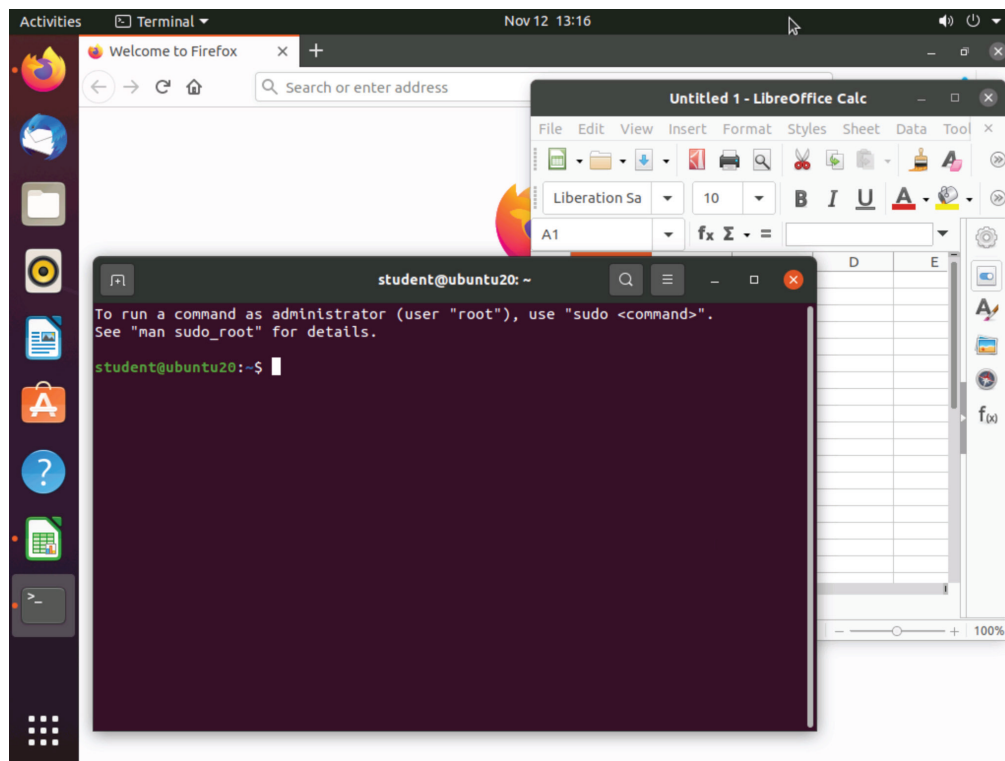
The Bash shell is covered in more detail later in this Lesson. It is the only shell covered by the CompTIA Linux+ exam objectives.

Common GUIs

Just as there are many different Linux distributions, there are also many different Linux graphical environments. Windows and macOS users have one GUI available to them—whatever graphical environment Microsoft and Apple choose to provide. Linux users have the freedom to install zero, one, or many GUI environments and switch between them.

These GUIs are usually distinguished by two characteristics: user-friendly interface and performance. Some users like the look and feel of a particular GUI over others. In addition, some GUIs consume more processor time and memory than others do. Luckily, many options are available in the Linux world.

Common GUI environments include **GNOME**, **KDE Plasma**, **Cinnamon**, and **MATE**.



Example of a GUI with running apps and menus.

Another important attribute of Linux GUIs is support for graphics-based applications, such as web browsers, presentation software, and image-editing programs. These types of software are critical to today's business environments and users.

Linux graphical interfaces provide many accessibility features that are worth exploring. Some of these include high-contrast displays, screen readers, magnifiers, visual alerts, and keyboard sticky keys.

Review Activity:

Linux Characteristics

Answer the following questions:

1. **Compare the advantages and disadvantages of GUI and CLI environments.**
2. **Explain how distributions differ from each other.**
3. **Why do servers tend to rely on CLI administration and desktops rely on GUI environments?**
4. **How might anyone contribute improvements to a piece of free and open-source software?**

Topic 1B

Understand Bash Interaction with Linux



EXAM OBJECTIVES COVERED

1.2 Given a scenario, manage files and directories.

Command-line administration relies on interfaces called shells. The default Linux shell is Bash, which has a particular syntax, or way of structuring commands. The syntax includes commands, command modifiers called options, and arguments. Bash also includes features such as tab completion and a history file. There are several common commands, directories, and applications available on most Linux systems, including the Vim and Nano text editors. Bash also supports privilege escalation. A solid understanding of Bash and its syntax makes Linux administration much easier.

Command Shells

The CLI is provided by software called a **shell**. The shell accepts user input, processes the input for syntax, and provides output back to the user. The default shell for most Linux distributions is **Bash**, and this is the shell that sysadmins should be prepared to work with.

Other common Linux shells include ksh, or KornShell, which is common among Unix servers; Zsh, or Z Shell, with quite powerful scripting capabilities; and Fish, or friendly interactive shell, an interface that provides a user-friendly experience and web-based configurations.

By way of comparison, Windows Server also uses shells: the traditional, DOS-like cmd.exe shell and Microsoft PowerShell. The current (at the time of this writing) default shell for macOS is the Zsh.



Bash is the Linux default and the only shell to concern yourself with for CompTIA Linux+.

Bash Characteristics and Syntax

Commands must be entered into Bash using a specific structure, or **syntax**. Each component of the syntax has a name to make it easier to understand.

The syntax components are:

- **Command:** The primary instruction given to the system.
 - **Subcommand:** A secondary, more detailed instruction supporting the primary command.

- **Option:** A command modifier that slightly changes the way a command is processed.
- **Argument:** The object on which the command acts. For example, in a command to delete a file, the argument is the name of the file to be deleted.

There are two basic forms, normal command and command-subcommand, to this syntax.

Normal Command Syntax

The normal command syntax relies on the three primary components of the Bash syntax: the command, options to modify the command, and an argument for the command to act upon.



Observe that there is a space between each of the three components!

As an example, here are several ways to use the list (`ls`) command with options and arguments.

Normal Command Syntax for the <code>ls</code> Command	Purpose
<code>ls</code>	List directory contents.
<code>ls -la</code>	List all (<code>-a</code>) directory contents in long format (<code>-l</code>).
<code>ls /var/log</code>	List the contents of the <code>/var/log</code> directory.
<code>ls -la /var/log</code>	List all contents of the <code>/var/log</code> directory in long format.



Most Bash error messages are descriptive, so be careful to read the error message to understand what went wrong.

Command-Subcommand Syntax

Many Linux commands support subcommands to specify particular information that the sysadmin needs. These commands rely on a different syntax from the basic format in normal command syntax. The sysadmin enters the primary command, then follows it with a space and a subcommand, and then a space and argument.

The `ip` command uses this format.

Command-Subcommand Syntax for the <code>ip</code> Command	Purpose
<code>ip addr</code>	Display all IP addresses for all interfaces.
<code>ip addr show eth0</code>	Display only IP address information for the <code>eth0</code> interface.
<code>ip help</code>	Display basic help about the <code>ip</code> command.
<code>ip link help</code>	Display help about the <code>ip link</code> subcommand.

```
$ ip addr show eth0
```

The `ip` command showing address information for the `eth0` interface.

Use Basic Bash Commands

There are many Bash commands. Some of the most often-used commands deal with file management functions, such as displaying files and file contents, moving from one directory (or folder) to another, or editing files.



It is customary in Linux to refer to folders as "directories."

Use Common Commands

The following commands exemplify the Bash syntax and enable users to begin working with the files and directories that make up Linux. These commands are used throughout this course and will quickly become familiar.

Command	Purpose	Example with Options	Result
<code>ls</code>	List the contents of the current directory	<code>ls /tmp</code>	List the contents of the <code>/tmp</code> directory
<code>touch</code>	Create a new empty file or update the timestamp on an existing file	<code>touch newfile.txt</code>	Create a new file named <code>newfile.txt</code>
<code>cd</code>	Change from one directory to another	<code>cd /etc</code>	Changes the current directory to <code>/etc</code>
<code>cat</code>	Display the contents of a text file on the screen	<code>cat data.txt</code>	Display the contents of the <code>data.txt</code> file
<code>less</code>	Display the contents of a file in windows that fit on the screen	<code>less data.txt</code>	Display the contents of the <code>data.txt</code> file screen at a time when the file would not normally fit on one screen
<code>tree</code>	Display the directory structure in a tree format	<code>tree /etc</code>	Display the subdirectories and files in the <code>/etc</code> directory in a tree structure
<code>shutdown</code>	Shut down the system	<code>shutdown -r now</code>	Restart the system immediately

Two common commands do not use options to generate an output. Use `whoami` to display the current user, and use `pwd` to display the present working directory.

```
student@ubuntu20:~$ ls
Desktop Documents Downloads Music Pictures Pul
student@ubuntu20:~$ pwd
/home/student
student@ubuntu20:~$ whoami
student
student@ubuntu20:~$ touch fileA
student@ubuntu20:~$
```

Command line interface showing the output of `ls`, `pwd`, `whoami`, and `touch`.



The number of Bash commands can be overwhelming. Start by using a few commands at a time, and make them a habit. The longer you work with Linux, the more comfortable you'll become with the commands.

Use Bash Tab Completion and History

Bash supports **tab completion**. Users can type in enough of a command to make it unique from any other command. Select the Tab key, and Bash automatically completes the command. This feature also works with file and directory names. Tab completion reduces typographical errors and increases speed at the CLI.

Bash also keeps a record of previously entered commands in a history file. This file can be referenced and used to repeat or edit commands.

The simplest way to work with history is by using the Up and Down Arrow keys. Select the Up Arrow key one time to recall the most recently used command. You can cycle through the command history by pressing Up Arrow or Down Arrow multiple times. Select Enter once the appropriate command is displayed.

Typing the history command displays the contents of the history file. Each entry in the file is numbered. Type ! and the command number executes that command.

```

11 pwd
12 whoami
13 touch fileA
14 clear
15 touch /hme/student/fileZ
16 touch /home/student/fileZ
17 ls
18 rm fileZ
19 ls
20 clear
21 touch /hme/student/fileZ
22 touch /home/student/fileZ
23 clear
24 history
25 pwd
26 clear
27 history
28 clear
29 history

student@ubuntu20:~$ !12
whoami
student
student@ubuntu20:~$

```

Retrieving a past command with the `history` command.



Some shells cache command history in memory. When the system is rebooted, the commands are no longer available. Bash writes the command history to a file stored on the hard disk. The commands are available even after multiple reboots.

Shell Tips and Tricks

Tab completion and history can make working in Bash far more efficient and less frustrating. Try to get comfortable using both features as quickly as possible, along with trying these other tips for easier use:

- **Tab completion:** Get in the habit of using tab completion for speed and to minimize typographical errors.
- **Use command history instead of rewriting long commands:** When you make a typographical error in a command or file name, do not manually retype 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.

Introducing Vim and Nano

Linux stores its configurations in text files. When a sysadmin needs to change system settings, these text files must be edited. There are many familiar text editors in GUIs, but what about Linux systems that do not have a graphical interface available, such as Linux server installations?

Two standard text editors exist that are run from the CLI and do not need a mouse or graphical interface: Vim and Nano. Here is a very brief overview of using these two editors.

Vim

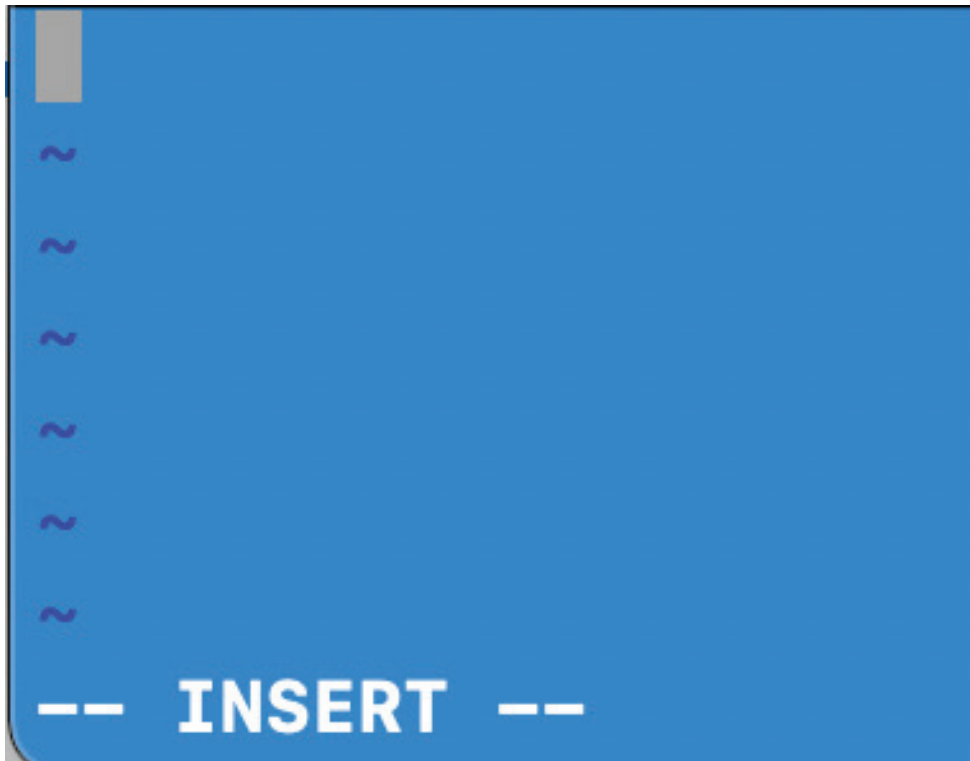
Vim is very powerful and complex. It uses three different modes, where each mode maps keyboard keys to different functions. For example, in **Insert mode** the keyboard acts as normal, inserting text into the file. If you're in Insert mode and type "abc," those three characters appear in the file's content. In **Command mode**, pressing a key on the keyboard issues commands to Vim instead of entering text in the file. Selecting the `i` key tells Vim to switch from Command mode to Insert mode. The third mode is **Execute**. This mode is entered by selecting the colon character, `:`, and it provides a command prompt to Vim where additional commands can be issued. For example, `:wq` places Vim in Execute mode, writes the files to the disk (save), and then quits Vim (`q`).

The many modes and commands can make Vim a little confusing. Strive to understand four basic functions: create/open, edit, save, close.

Function	Command	Result
Create/Open	<code>vim filename</code>	Create a new empty file, or open an existing file in Vim.
Edit	<code>i</code>	Enter Insert mode, and begin making edits.
Save	<code>ESC</code> and <code>:w</code>	Move out of Insert mode and into Command mode, and then save (write) changes.
Close	<code>ESC</code> and <code>:q</code>	Move to Command mode, and then exit.



For efficiency, type `:wq` to combine the save and close options.



The Vim editor. Note the INSERT flag in the lower left corner.



Vim is covered in more detail in Lesson 5. A summary is provided here to permit immediate editing of text files if necessary.

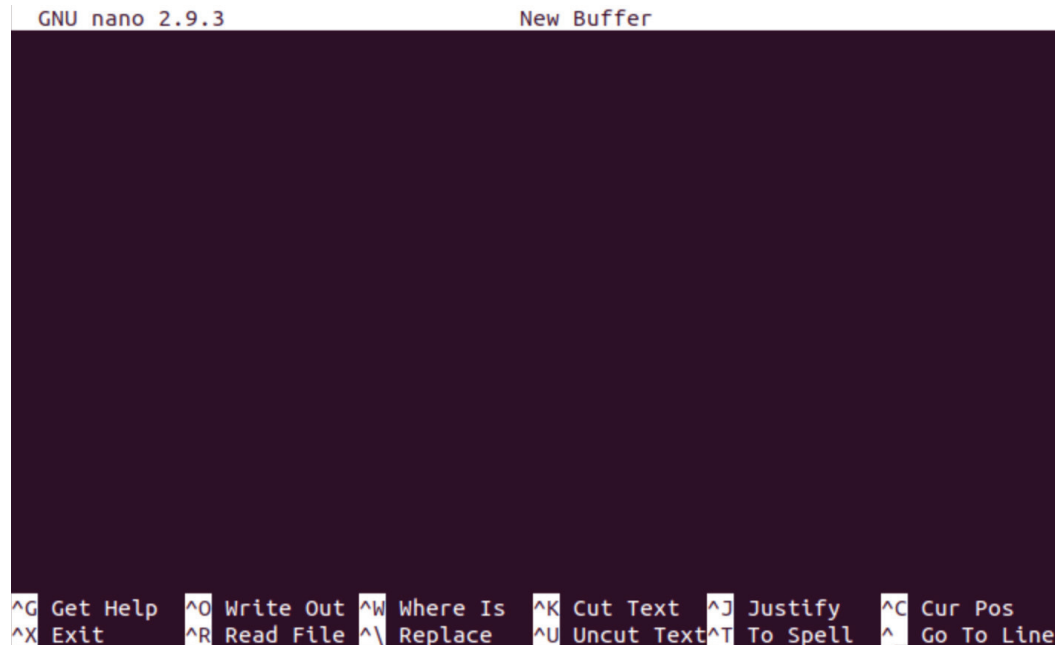
Nano

Nano is a popular and common alternative to Vim. It's simpler but less powerful. However, in many cases sysadmins don't need the power offered by Vim, which makes Nano a useful choice for basic editing functions.

Nano does not have modes. Pressing keys on the keyboard inserts text into the file, just as expected with most editors. To save and close the file, use keyboard shortcuts using the `Ctrl` meta key. For example, `Ctrl+O` saves the file, and `Ctrl+X` exits the file. You may have used similar keyboard shortcuts in other applications.

As with Vim, it's critical that you are capable of using Nano to create or open, edit, save, and close files.

- Type `nano filename` to create a new empty file or open an existing file with Nano.
- To edit the file, simply begin typing. Use the arrow keys to move the cursor.
- Type `Ctrl+O` to save changes.
- Type `Ctrl+X` to exit Nano after saving changes.



```

GNU nano 2.9.3                               New Buffer

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line

```

The Nano editor. Note the menu at bottom of the window.



Nano is covered in more detail in Lesson 5. A summary is provided here to permit immediate editing of text files if necessary.

Some Linux distributions install both Vim and Nano by default, while others will include only one or the other. It is essential for you to be able to use both editors at a very basic level (open, edit, save, close) so that you are capable of editing files with whichever tool is available.

Introducing su and sudo

There are three types of accounts on Linux systems: root, standard user, and service.

The administrator account in Linux is called root. Logging in to the system with administrator access is frowned upon. The security best practice is to log on with a standard **user account**, and then, if necessary, switch your user account to root. The command to accomplish this is `su`.

Type `su root` to switch from the standard user to root.

Type `exit` to leave the root user and return to the standard user.

Type `su - root` to switch from the standard user to root with the root profile. Note that there is a space on each side of the dash character. Again, type `exit` to close the root login and return to the standard user account login.

```
student@ubuntu20:~$ whoami
student
student@ubuntu20:~$ su - root
Password:
root@ubuntu20:~# whoami
root
root@ubuntu20:~# exit
logout
student@ubuntu20:~$ whoami
student
student@ubuntu20:~$ █
```

Elevate privileges from standard user to root, and confirm the change with the `whoami` command.

The problem with the `su - root` command is that it grants all administrative privileges to the escalating user, assuming the user knows the root password. In other words, the user is either a non-privileged account with almost no administrative authority or the full root user account with all possible administrative authoring—and nothing in between. Sometimes, administrators want to delegate specific, defined activities that require root authority, but only those particular activities.

Sysadmins can edit a file named `/etc/sudoers` to delegate specific tasks to individual users and groups. The specified identity may exercise that task as if they are root, but nothing else. This is a much safer alternative than giving full root privileges to individuals who may not be fully qualified to run the system. This delegation concept is critical to good security.

To accomplish a delegated task, simply precede the command with `sudo`. You will usually be prompted for your password and given a warning to be careful on the system. The command then executes.

As a security measure, some distributions disable the root user account and force the use of `sudo` on specific user accounts.

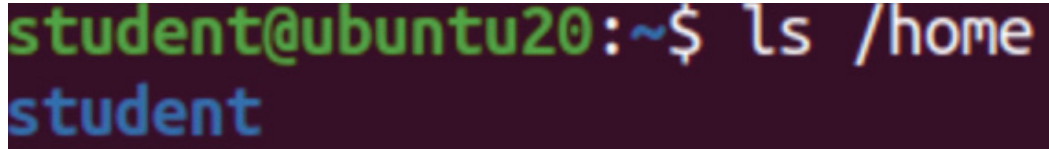


Privilege escalation using `su` and `sudo` are covered in more detail in a later Lesson. A summary is provided here in case it's needed for hands-on activities.

Identify Common Directories

With so many Linux distributions available, administrators rely on the Filesystem Hierarchy Standard to understand the default location of particular resources. There are three common directories that administrators work with on a regular basis.

- **/home/username:** Each standard user has a specific and private directory used to store personal files, profile settings, and other data. These user directories are subdirectories of /home.
- **/etc:** Most system configuration files are stored in the /etc directory.
- **/var/log:** Log files for the system and applications are stored in the /var/log directory.

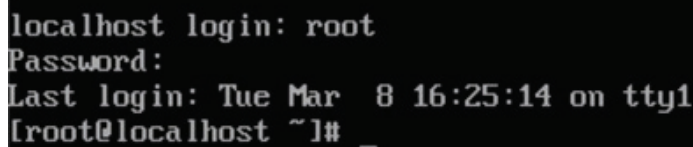
A terminal window with a dark background. The prompt is 'student@ubuntu20:~\$'. The command 'ls /home' has been entered, and the output 'student' is displayed on the next line.

Use the command `ls /home` to display a few existing user directories.

There are many other standard directories, and they are covered in a later Lesson.

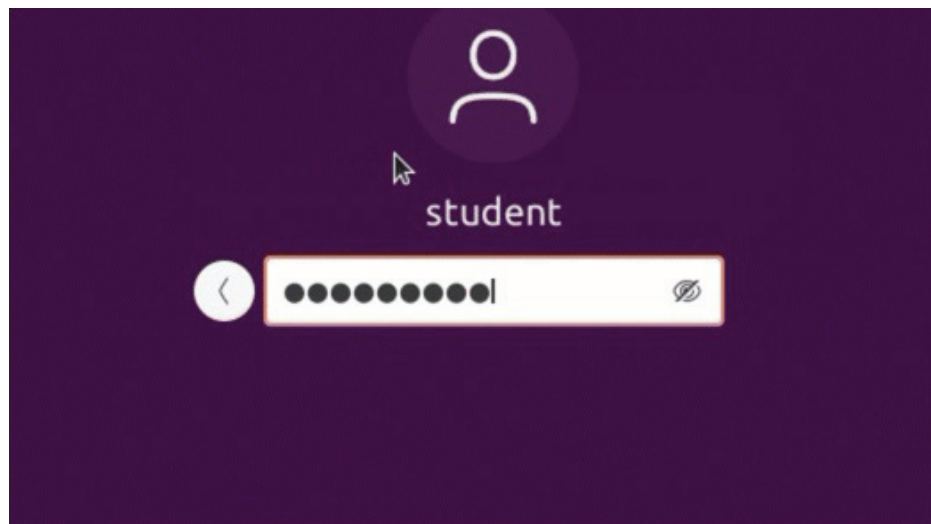
Log In Using the GUI and CLI

If the Linux system is configured to boot to the CLI (or doesn't have a GUI at all), users are prompted for a username and password. After entering these two values, the system authenticates the account and loads profile settings from files stored in the user's **home directory**.

A terminal window with a black background. The text shows a login sequence: 'localhost login: root', 'Password:', 'Last login: Tue Mar 8 16:25:14 on tty1', and the prompt '[root@localhost ~]# _'.

Logging in from the command line.

If the system boots to the GUI, a login prompt is displayed that may show available user accounts. A password is entered, and then the user is authenticated. Profile settings related to the GUI—such as desktop backgrounds and menu items—are then loaded.



Logging in from a GUI.

Review Activity:

Interact with Linux

Answer the following questions:

1. **An administrator asks you to make a change to the system's configuration. Why would you need to use Vim or Nano to accomplish this task?**
2. **What types of files will be found in the /etc directory?**
3. **Explain the difference between the su and sudo commands.**

Topic 1C

Use Help in Linux



EXAM OBJECTIVES COVERED

This topic covers the foundational concepts of documentation and does not address a specific exam objective.

Linux includes strong built-in documentation in the form of manual pages, which provide brief explanations of the command, available options, and a few examples. They are short, concise references. It is essential that sysadmins become comfortable accessing and using manual pages. More significant documentation may be built into applications, stored on the system, or available online at vendor websites.

Linux Documentation

There are several ways of getting help in Linux. The most common are the manual pages, referred to as “man pages” for short. There is built-in documentation for the system and some applications, too. Many online resources also exist, and they are often the most up to date.

Because there are so many commands, and because each command has so many options, it’s very common to use the man pages as a quick reference for displaying the available options.

Manual Pages

It’s common for new Linux users to ask for help and then be asked, “Did you check the man pages?” That’s because man pages are the primary reference for standard Linux commands. The man pages provide syntax information and usage examples. Perhaps most important, the available options are displayed. Because of the number of options for each command, and the fact that many options differ from command to command, the man pages provide an essential quick reference.

The syntax for using man pages is `man {command}`.

For example, to display help for the `ls` command, type `man ls`.

```
student@ubuntu20:~$ man ls
```

Obtaining help for the `ls` command, using the man pages.

Man Page Sections

Man pages are organized into eight sections. A section defines what category the command belongs to. These sections are displayed by a number following the command. For example, `fsck (8)` indicates that the `fsck` command is found in section eight.

```

FSCCK(8)

NAME
    fsck - check and repair a Linux filesystem

SYNOPSIS
    fsck [-lrsAURTMNP] [-C [fd]] [-t fstype] [fil

```

The man page for `fsck`.

Note that the numbers in this list are man page section numbers, not merely part of the list. In other words, “System calls” is in **Section 2** of the man pages documentation, not just the second item in this grouping.

Section eight is probably the most commonly used section for administrators.



It is not necessary to memorize the section numbers. Sometimes it can be useful to recognize what section might contain the man page for a given command. If you're interested in a deeper dive into the conventions and configurations of man pages, take a look at the [man-pages project website](#).

man Page Navigation

You can use several different keys to navigate through the man pages, all mapped to specific actions.

`Home` -- Move to the beginning of the man page.

`End` -- Move to the end of the man page.

`Page Up` -- Scroll up one page.

`Page Down` -- Scroll down one page.

`/` -- 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` -- Exit the man page, and return to the shell.

For example, to search for the string “directory” in the `ls` man page, open the man page and then type a forward slash character and the word “directory.”

```

$ man ls
/directory

```

The first line opens the man page for `ls`, and the second line searches the keyword `directory`.

Built-In Documentation

Most commands include help references. Add the `-h` option, or `help` after the command to display this reference material.

The `what is` command provides a brief description of the specified command.

The syntax for `what is` is `what is {command}`.

Finally, built-in documentation can be found at `/usr/share/doc`. This directory contains some Linux and application help files. Not all applications store documentation at this location, but it's worth checking.

Online Documentation

There is a great deal of information available online that covers Linux administration, applications, security configurations, and network services. This documentation may be provided by vendors, community groups, online forums, article repositories, and other sites.

- **Linux distribution vendors:** Vendors such as Red Hat and Ubuntu have large repositories of reference information.
- **Linux application vendors:** Vendors for products such as Apache web server, Vim, and Firefox provide many references for their applications.
- **Linux Documentation Project:** This is a community project dedicated to providing documentation for Linux, including how-to documents, man pages, and guides. These references are found at <https://tldp.org/>.

Review Activity:

Help in Linux

Answer the following questions:

1. **Name three things a man page might provide a user.**
2. **Why might vendor websites be the best source of information about an application or service?**

Topic 1D

Identify the Linux Troubleshooting Methodology



EXAM OBJECTIVES COVERED

This topic covers the foundational concepts of troubleshooting and does not address a specific exam objective.

One of the primary skills and duties of a systems administrator is to troubleshoot problems with servers, the network, and data access. It is important to have a methodology for troubleshooting. You should also recognize that troubleshooting methods may change by situation, skill level, and experience with the network environment.

Troubleshooting Methodology

A formalized and consistent **troubleshooting methodology** can make identifying issues and discovering fixes more efficient. While the steps can vary depending on the actual issue and components involved, there are several universal troubleshooting steps.

The following list represents the basic steps in a troubleshooting methodology:

- Identify the problem.
- Determine the scope of the problem.
- Establish a theory of probable cause/question the obvious.
- Test the theory to determine the cause.
- Establish a plan of action.
- Implement the solution or escalate the issue.
- Verify full system functionality.
- Implement preventive measures.
- Perform a root cause analysis.

Throughout the process you will find it helpful to document findings, actions, and outcomes of the various steps.

Identify the Problem and Determine the Scope

Identify the Problem

The first troubleshooting phase is to identify the problem. The problem may be discovered for you by the end users you support, exposed by log files, identified by monitoring software, or indicated by lights on the server. There are many ways through which the problem may be detected. Once a problem is identified, a service desk ticket is used to track it.

Determine the Scope of the Problem

Once a problem is identified, gather additional information to determine the scope of the problem. Start this process by asking users for additional details or examining log files. Attempt to replicate the problem by asking users to show you what they were doing when the problem was encountered or to try to recreate the situation where the problem first arose. It is a good practice to back up data if there is any risk to the data during the troubleshooting phase. You must use your own judgment as to whether a data backup is necessary before you begin troubleshooting. Finally, consider whether you have the skills to address the problem or if you need to escalate the service desk ticket to another administrator.

One of the most important steps is to determine whether the problem exists on only one server or on multiple servers. The scope of the problem could be hardware based and, if so, may be isolated to that device. It could be network based, in which case, multiple devices may be affected. It could be software based, such as a misconfiguration or a bug. This also may impact multiple servers.

For example, if one workstation cannot access a file server, but all other workstations can, the problem likely lies with that workstation. If many workstations cannot access the file server, the problem likely lies with that server or with the network between the workstations and the server.



In Linux, the log file service is named "rsyslog." Services are covered in Lesson 9.

Establish and Test a Theory of Probable Cause

Establish a Theory of Probable Cause: Question the Obvious

The next troubleshooting phase is to establish a probable cause for the problem. It is essential to keep this step as simple as possible. Troubleshooting often begins with very basic steps, such as confirming that the system is plugged in and powered on. More complex problems may require you to examine log files, talk to users or other administrators, or check the hardware.

When troubleshooting, identify any common elements or similar problems that might span multiple servers or network devices. Such common elements might include a new or updated piece of software, a new device driver, or a new configuration.

Check for any recent changes to the environment. These changes may have been implemented by another IT staff member or a stakeholder, such as a manager or other user. Recent changes are common culprits for issues.

Test the Theory to Determine the Cause

Next, test the theory by verifying that the likely cause is indeed the culprit. This phase involves research or other testing. Very simple problems may actually be solved during this step. If your theory is confirmed, then move on to the next phase, which is to establish a plan of action. If your theory is not confirmed, then you must establish and test a new theory.

Establish and Implement a Plan of Action

Establish a Plan of Action

The plan of action for addressing the problem must recognize that service interruptions and data loss should be avoided. If a server needs to be brought

down to replace hardware, or if data has been lost due to a HDD failure, the end users must be notified. The plan of action defines the steps to be taken. These steps should be defined ahead of time rather than created during the implementation of the solution. It is useful to provide the impacted users with an expected duration of the outage.

Implement the Solution or Escalate

In this phase, follow the plan of action established earlier. It is important not to deviate from the plan. You may not have the knowledge to implement the plan and need to escalate the problem to the vendor's support team or other members of your own team.

When following a plan of action, be sure to only make one change at a time, and then test the result. If you make multiple changes simultaneously, it is difficult to identify exactly which change corrected the problem. If a given change does not solve the problem, reverse that change, and then try another option.

Verify, Prevent, Analyze, and Document

Verify Full System Functionality

Once the potential solution has been implemented, the next phase is to test for functionality. Your goal is to ensure that the server has returned to the service levels that are defined by the system parameters. The server performance baseline that you performed during the deployment portion of the server lifecycle will be very useful as a comparison.

Implement Preventive Measures

It may be possible to preemptively reconfigure other servers to avoid a repeat of the same problem. It may also be possible to implement additional technologies (such as a redundant array of independent/inexpensive disks [RAID]) or additional practices (such as backups) to prevent future instances of failure. In some cases, additional training or documentation may also be necessary.

Perform a Root Cause Analysis

Once service is restored to your users, it is time to evaluate why the problem occurred. Identifying the root cause permits you to change processes or implement different technologies to avoid the problem in the future.

Document Findings, Actions, and Outcomes Throughout the Process

Documenting the symptoms of the problem, the results of research into potential solutions, and the results of each step of the plan of action (whether the step was successful or not) permits you to understand your environment better and therefore helps to prevent possible future problems. Note that documentation is not a separate step but rather a good practice used during each phase of the troubleshooting process.



Some service desk management software requires the use of tickets. Such software may require that troubleshooting documentation be entered before the ticket can be closed.

Review Activity:

Troubleshoot in Linux

Answer the following questions:

1. **A user contacts you to find out why they cannot access a directory. Using the troubleshooting methodology, how would you narrow the scope of the problem?**
2. **When should you escalate a problem?**
3. **True or False? Documentation should be created only at the end of the troubleshooting process.**

Lesson 1

Summary

By understanding how open-source licensing allows many Linux distributions to exist, it is easier to differentiate between Linux and other operating systems. Linux is widely used with various hardware, but one of its most common roles is as a server OS. Servers need to maximize the use of available hardware resources, so Linux is often managed via the Bash shell rather than a graphical interface. Bash is more hardware-efficient and allows tasks to be scripted and scheduled.

Command-line administration has its challenges, however, especially when managing system configurations through text files. Applications such as the Vim text editor are used to edit these files, resulting in updated settings. Text editors and Bash commands may be difficult to remember and often rely on one or more options to modify their behavior. Administrators use built-in man pages to reference command functions and options. Often, such references are made as part of a larger troubleshooting methodology that attempts to provide comprehensive and effective problem-solving.

Guidelines

These best practices and guidelines are provided for your use as revision tools or as quick references in your job role.

- **FOSS:** Recognize and describe free and open-source software, including advantages and disadvantages.
- **GPL:** Understand how the GPL influences the development and availability of the Linux OS.
- **Distributions:** Understand what a distribution is and how distributions differ from each other.
- **GUI vs CLI:** Understand the advantages and disadvantages of each environment.
- Use Vim and Nano to open, edit, save, and close files.
- Recognize both command syntax structures:
 - command -option argument
 - command subcommand argument

Command Reference Table

This list of commands and their associated syntax can also be found in Appendix B.

Command	Syntax	Purpose	Covered in
<code>ls</code>	<code>ls [option]</code>	List the contents of the current directory.	Lesson 1, Topic B
<code>cat</code>	<code>cat [file-name]</code>	Display the contents of a text file on the screen.	Lesson 1, Topic B
<code>cd</code>	<code>cd /etc</code>	Change from one directory to another.	Lesson 1, Topic B
<code>pwd</code>	<code>pwd</code>	Displays the present working directory.	Lesson 1, Topic B
<code>whoami</code>	<code>whoami</code>	Displays the username of the current user.	Lesson 1, Topic B
<code>touch</code>	<code>touch [file-name]</code>	Create a new empty file or update the timestamp on an existing file.	Lesson 1, Topic B
<code>man</code>	<code>man [command]</code>	Display manual, or help, pages for a specific command.	Lesson 1, Topic C
<code>whatis</code>	<code>whatis [command]</code>	Provides a brief description of the specified command.	Lesson 1, Topic C