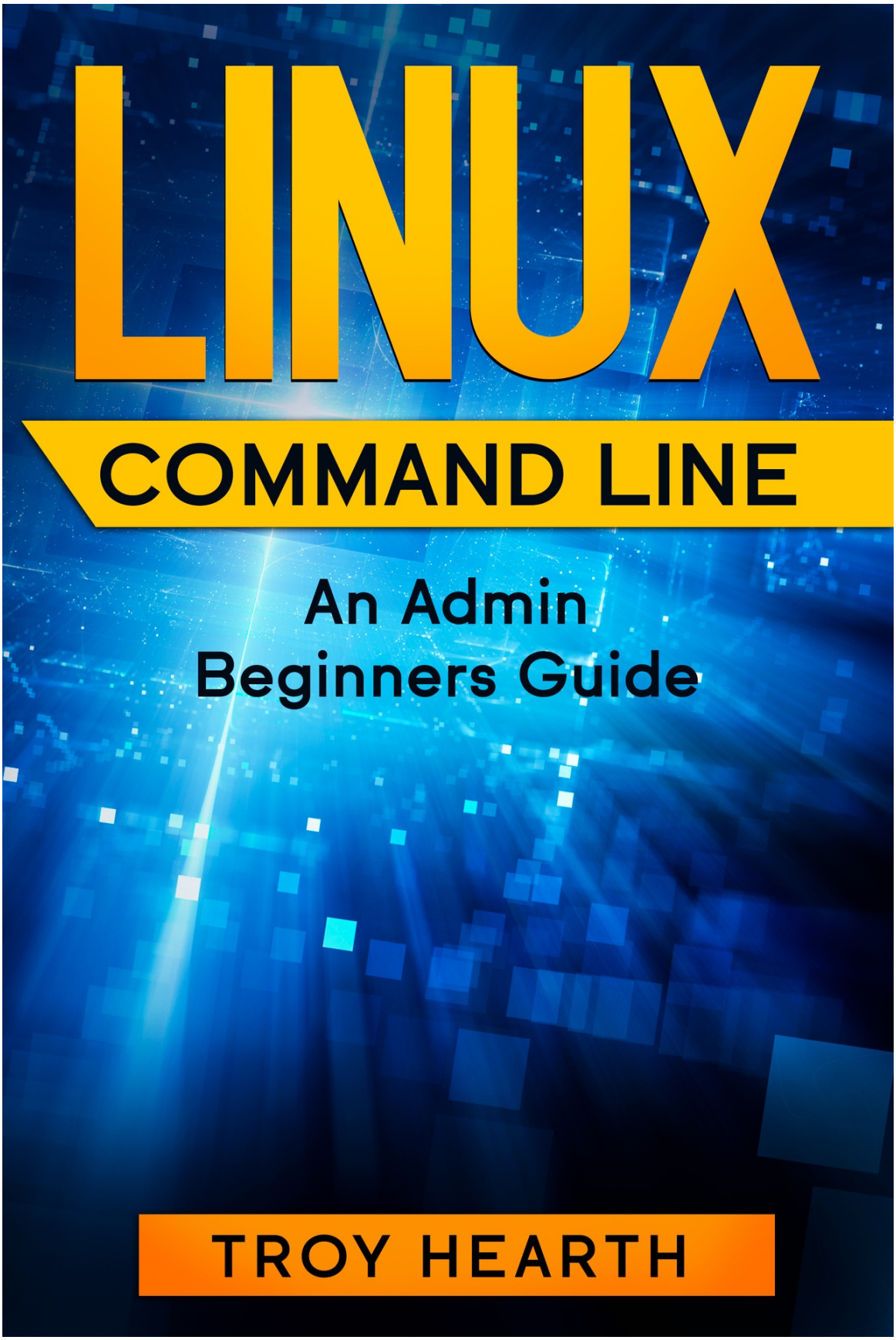
LINUX

COMMAND LINE

An Admin Beginners Guide

TROY HEARTH



LINUX COMMAND LINE
An Admin Beginners Guide

Disclosures

by Osprey Innovations, LLC

While all attempts have been made to verify the information provided in this publication, neither the author, nor the publisher assumes any responsibility for errors, omissions, or contrary interpretations on the subject matter herein. This book is for informational purposes only. The views expressed are those of the author alone and should not be taken as expert instructions or commands. No guarantees for earnings or any other results – of any kind – are being made by the author or publisher, nor are any liabilities being assumed. The reader is entirely responsible for his or her own actions.

Adherence to all applicable laws and regulations, including international, federal, state and local governing professional licensing, business practices, advertising, and all other aspects of doing business in the US, Canada, or any other jurisdiction is the sole responsibility of the reader or purchaser. Neither the author nor the publisher assumes any responsibility or liability whatsoever on the behalf of the purchaser or reader of these materials. Any perceived slight of any individual or organization is purely unintentional.

Table of Contents

CHAPTER ONE: INTRODUCTION

Introduction

If you want to learn how to use the Linux operating system, but felt it was too complicated, this book is for you. I'm confident that as you read this book and practice the exercises that you will be able to develop a working knowledge of the Linux command line.

Once you develop a working knowledge of the command line, you will be able to tackle more advanced Linux projects with confidence. Maybe you've wanted to program your own server system, adapt a Linux system for home use, or just dabble in computer programming. Whatever the reason, you're likely to find this book helpful for your situation.

How This Book is Organized

You do not necessarily need to read this book cover-to-cover if you already understand some of the concepts. Below is a brief list of the chapters covered, along with their contents:

Linux Overview

This chapter gives a brief overview of the Linux operating system, how it was invented, what exactly open-source software means for you (the end-user), and various types of Linux and how they may differ from each other.

Linux Shells

The shell provides one way for you to communicate with your machine via a GUI or LUI interface. If you aren't familiar with these terms or want a quick refresher, read this chapter.

Basic File Directory

This chapter covers the names of basic files found in Linux, such as bin and root, and what they do.

The Linux Command Line--The Basics

If you believe you are ready to just jump in and start learning Linux commands without any preliminary information, you may skip the other chapters and start

here. This chapter covers some basic Linux commands, including cd and pwd. This chapter also covers basic syntax, which will allow your commands to read correctly.

The Linux Command Line--More Advanced Commands

This chapter contains more advanced commands, such as ping, tar, and locate.

Linux System and User Management

Learn how Linux manages users and protects its important files by only granting administrator access to users with the proper permissions.

Understanding Linux Security

Even though it's open-source, Linux is one of the most secure operating systems available. Learn why and how to keep it that way in this chapter.

Suggestions for Continued Learning

Once you've gotten the basics down, maybe you would like some suggestions on what to learn next. This chapter will give you ideas on how you can continue learning about Linux.

IMPORTANT :

chapters in this book contain a lot of dense information. It is highly recommended that you read these chapters with a pen and paper in hand and also have access to a computer where you can practice and get comfortable with the commands. The purpose of this book is to help you get comfortable with the Linux command line so you can more confidently use Linux's most powerful features. It is not a casual read, but rather, a beginner's instruction to the basics of the Linux command line.

CHAPTER TWO: LINUX OVERVIEW

What is Linux?

Linux is an operating system or OS. An operating system is defined by Wikipedia (https://en.wikipedia.org/wiki/Operating_system) as follows:

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

Think of the operating system as a body with the Linux kernel as the brain. The brain uses arms, legs, fingers, toes, etc (computer hardware) when needed. The brain plans out when to take lunch and other activities (software resources) during the day. Think of software resources as actions such as waking up, brushing teeth, and going to work. The brain also provides common functions needed for living such as breathing, heartbeat, decision making, etc. The Linux kernel needs to perform these common services such as incrementing the internal clock, etc.

The computer hardware can be thought of as the mouse and keyboard and video display. The operating system manages the input from the keyboard and mouse and displays the output on the screen. A software resource such as Word or the app running on a Kindle just sends what it wants to display to the operating system. The OS manages the hardware that changes the pixels to make the picture on the display. The OS also gives each running program a piece of time to run otherwise only 1 program could run at a time. Can you imagine only being able to check the internet, then shutting Firefox down so that you could check email, then closing that to open the calendar app?

Why Use Linux?

You may have noticed the popularity of the Linux Operating System (or Linux OS) in webserver work and other backend computer functions. It has a reputation for being dependable and having long up-time which means it doesn't crash often or require frequent reboots. A typical website may be running on a Linux Operating System that has continued to run uninterrupted for months or years without a restart.

Linux is one of the most secure, robust, and user-friendly free open-source operating systems available. The Linux OS can easily compete with Microsoft's Windows and Apple's iOS in terms of features, which is truly impressive for free

software maintained by a population of volunteers.

Linux is open-source and there are several versions that are completely free. What is open-source? Open-source means that anyone can download and look at the computer code that runs Linux and build it to generate their own version of Linux. Because Linux is open-source, there is an entire ecosystem of volunteers that are constantly fixing and adding features to Linux.

There are paid versions of the Linux OS that offer technical support for a fee. Many corporations use these companies to keep their servers running smoothly and to also keep their systems secure.

The Linux operating system also has applications for home use. However, one disadvantage Linux has compared to Windows and Apple OS is the lack of infrastructure. Popular tools such as Word, Excel, and Powerpoint are not supported on Linux although there are good alternatives such as LibreOffice or Google Docs. Although it cannot run all the apps and programs that iOS and Windows can, depending on what you want to do, a Linux operating system may still fit your needs. Chromebooks, for example, run on a Linux kernel.

A Linux system is great for learning computer programming because many programming tools are already installed or are very easy to install. Also, many people prefer Linux when building computers because they can view and modify its source code to suit their needs.

It's also one of the smallest and most portable operating systems available. It's the operating system of choice for Raspberry Pi, an inexpensive mini computer that you can hold in the palm of your hand, and is also the system of choice for serious programmers, backend developers, and security-conscious end users.

How Linux Started

As mentioned earlier, Linux is open-source software modeled after the AT&T proprietary software UNIX, which stands for Uniplex Information and Computing Service. Linux, although it shares many features with UNIX, does not contain any of the UNIX source code. It was created from scratch to be compatible with UNIX, so people who were used to UNIX could more easily transition to Linux.

The creation of Linux as we know it today is credited to a developer named **Linus Torvalds** , who created the Linux kernel from scratch, then made it completely free and open-source, even citing in the licensing agreement that it was not to be sold. Later, Linux was relicensed under the GNU GPL license,

enabling it to be sold under certain conditions. The first version of Linux available to home users came out in 1992.

Linux developers focused on keeping Linux "free" and open-source rather than allowing it to be changed and claimed by corporations like Microsoft. This resulted in Linux taking some interesting turns in its evolution. Rather than adding various apps and accessories for home users, Linux evolved to be a more streamlined and clean-running operating system, embraced by professionals and programmers, especially for use in servers and backend development.

Linux is licensed under a GPL version 2.0 license. Referred to as "**copyleft** ," this software can be used, modified, and distributed freely, under the condition that any derivative works produced from it are also released as open source.

What is Open Source Software?

Open-source software is software that allows you to see the source code. This allows users to potentially read and create similar programs, as well as build add-ons for existing programs. A common misconception is that all open source software is free for all uses; this is not necessarily true.

All open-source software is not necessarily completely free software, in that it costs no money, as some open-source software is only free for home and personal use. Open-source software is also sometimes sold at distributors, though it does not necessarily give you the right to modify or re-distribute the software. Other open-source software is free to download, but it charges a contract fee for licensing.

All open-source software is free, however, in that it is intended to help further people's understanding and enthusiasm for programming. Being able to view the code and not allowing certain Linux code to become proprietary trade secrets has helped Linux develop a robust community of enthusiastic developers and users.

Linux software, in comparison to most other open-source software, is free in both its open-source characteristics and the lack of financial cost to the end-user. It can also be modified in any way the end user wishes. This kind of software is called **Free Open Source Software** or **FOSS** . The status of FOSS does not necessarily apply to all open-source software. Keep in mind that some parts of the Linux software are not FOSS.

Is Open Source Software Safe?

Someone familiar with Windows might think that with free software the fact that

anyone can create, edit, and update is fairly risky. However, Linux edits go through an extensive review process before being accepted and therefore Linux is one of the most secure systems available, perhaps even more secure than the popular, security-based Apple OS. Part of this is due to the Linux operating system, which functions a lot like the Apple OS, only allowing certain approved programs to be downloaded and only giving downloaded programs limited permission to interact with the rest of the system.

Linux also has a relatively small user base compared to the other operating systems and isn't usually a target due to the extra work it would take to hack into it. In fact, it's one of the systems security-conscious individuals such as IT professionals usually choose to have on their computers.

Being open-source has been useful for security because many developers are constantly examining the code for vulnerabilities and quickly fixing them.

In contrast, closed-source kernels, such as Windows and Apple, rely only upon the expertise of the developers working for Microsoft or Apple. Patches for bugs are released occasionally. Other times, minor bugs in software and even significant security concerns are overlooked until there are serious customer complaints. This is one of the downsides to having to depend on a handful of programmers to maintain a closed-source product.

With Linux and a little bit of training, you can spot bugs and weaknesses in programs, then report or even attempt to fix them yourself. Windows and Apple simply do not grant the end-user that level of control. Since a lot of the people who keep Linux running may be security-conscious individuals who depend on Linux to keep their company machines in working order, security concerns may often take top priority.

The Linux Community

Dedicated volunteers put in hours of their time creating and helping service new programs that are comparable and even compatible with software written for commercial operating systems or hardware. This is a huge undertaking and unlike commercial developers like those who work for Microsoft, Google, or Apple, the people responsible for new Linux programs are just as likely to be hobbyists who program computers on the side or do something unrelated to computer programming as skilled programming professionals.

It's important to keep these things in mind if you are working with a Linux system. Rather than a product mass-produced by the commercial industry, certain distributions of the Linux operating system are comparable to a carefully

crafted homemade product. It may take some tinkering to get them to work the way you want and you could choose a version by mistake that isn't made to do what you want it to do.

Which Distribution Do I Need?

Just like other products, commercial and otherwise, distributions of Linux, called **distros** , range in quality. The slightly older distros of Linux tend to be more reliable and have more support than the newer, "bleeding edge" versions of Linux. It's easier for new OS developers to build off of existing Linux kernels than it is for them to create a brand new operating system from scratch. As a result, most Linux operating systems are fairly similar to one another, and a lot of existing Linux software can be run on "new" versions of Linux.

Although there are several versions of Linux available, they just tend to be different versions or "flavors" of a few main versions of Linux. This list includes Debian, Fedora, and Ubuntu.

What to Look for in a Distro

When choosing a Linux distro, you might want to consider your own needs, including how comfortable you are with the command line, what purpose you need the distro for--do you need it for headless IoT or are you trying to set up a home server or desktop? The answers to these questions may greatly influence which distro is right for you; Linux distros are not necessarily one-size-fits-all.

What Do You Need Linux For?

Home Use

For home use or programming practice, it is probably best to use Ubuntu. It's fairly stable and comes with a lot of features out-of-the-box, including a fully functional desktop environment and package manager that makes it behave similarly to an Apple OS or Chromebook. Because of its familiarity, it might be less intimidating than some of the other versions of Linux available.

It is also a widely used version of Linux with lots of community support. Community support is important if you run into problems because a problem in Ubuntu has probably been answered.

The downsides of almost any version of Linux for home use is that not all of it will function like an Apple or Windows machine. Although Linux is more popular than Windows or iOS for server applications, it hasn't taken off in the

consumer market. As a result, most software companies haven't created Linux versions of their products.

Photoshop, for example, does not have a Linux version. However, there are free open-source alternatives such as GIMP for Linux. Plus, Linux has a surprising number of programs that are made specifically for the Linux OS. Unless you need something specific like Microsoft Office, you may be able to find a decent work-around or alternative software.

Server or IoT

Some people are more concerned about programmability and stability than ready-installed or ready-to-install packages. In this case, a "headless" version of Linux, streamlined and without all of the "bells and whistles" such as a desktop environment to get in the way, might be more appropriate.

In this case, consider the Fedora server version for security, stability, and support. If you run more than just a home server or your business or product relies on Linux to always function properly, you may want to opt for a product backed by Red Hat.

Project Learning

Debian might be a good idea if you are trying to learn robotics or remote programming. If you are experimenting with a Raspberry Pi, the Raspbian operating system built especially for Raspberry Pi, might not be a bad distro to try either.

If you are a more advanced tinkerer, you may want to look at a basic flavor of Arch Linux or a similar distribution that is lightweight and flexible while still being relatively well-supported. It all depends on the project.

You should choose and take advantage of a distro with an active and robust community of hobbyists and enthusiasts for any projects you are trying to do. Internet research, Youtube videos, and community forums may also help you decide which distro is right for you and your project.

Chapter Summary

- Linux is an open-source operating system developed in 1991. Linus Torvolds is credited with creating the first functional Linux kernel that was first ported to home computers and eventually distributed worldwide. He is also one of the developers that started the tradition

of making Linux FOSS, which stands for free and open-source software.

- Eventually, Linux developers agreed that it was okay for it to be sold but still remain open source, and Linux was relicensed under GNU GPL.
- As Linux became more popular, other versions that were friendlier to home users, such as Ubuntu, were developed.
- It may not seem like a big deal to most casual computer users that the Linux project exists but it's potentially a game-changer for programmers. It's existence also means that security-conscious users don't have to put up with data-mining companies, like Google and Microsoft, just to use a computer.
- The Linux vision continues to provide everyone with the freedom to develop and to learn.

CHAPTER THREE: LINUX SHELLS

What is the Linux Shell?

In Linux, a "shell" is a command interpreter. Put simply, the **shell** is a program that helps the user to communicate with the rest of the operating system. Shell programs make it much easier to communicate with the machine and tell it what to do.

Brief History

Once upon a time, computers were massive machines that took up a large part of a room. They didn't have screens, keyboards, or mice; instead, users communicated with them through manually feeding them instructions on paper cards by using teletype machines or a series of switches. Even though these machines were massive, they could only run one program and the program instructions had to be re-fed to the machines every time they were turned on or a new program was needed.

Someone realized that computers could do multiple tasks if they dedicated 30% of the time to Paul, 30% of the time to Betty, and 40% of the time to Pat. The computer could run Paul's program for 30 seconds then Betty's program for 30 seconds, then Pat's program for 40 seconds. One of the first multi-tasking operating systems was named Unix. As discussed above, the method of communicating with the operating system using a shell which evolved from the old paper card programs.

Over time, computers started getting smaller but the concept of how to communicate with them, along with a lot of the terminology carried over from the early days stuck, including the idea of shells and terminals. Originally, the **terminal** was a teletype that looked like a typewriter and display. The terminal provided a more convenient way to feed information to a computer while the shell provided a way to process the keyboard inputs.

Eventually, developers who wanted more user-friendly machines and to get computers in the hands of more than just programmers created less complicated operating systems where the work of the shell and its terminal could be hidden from the user. Thus, operating systems with desktop environments were developed.

Back to Shells

The shell of an operating system is a program that takes input from the keyboard and gives it to the operating system as commands to perform. The shell communicates with the terminal that the user uses to interact with the operating system.

In a desktop environment, the shell communicates works without being seen. If it's a **graphical user interface** (or **GUI**) all the user has to do is navigate to certain icons or click on certain buttons to get the machine to perform tasks or open programs. GUIs aren't the only means of communicating with the shell, however.

The shell also communicates with the **line user interface** , or **LUI** , which can also be referred to as the **command line interface** , or **CLI** . An example on Windows is the Windows Command Prompt, which is the black screen that prompts the user to enter code so the computer can perform certain tasks.

Although you can run programs using a GUI in Linux, using the LUI to execute programs is far more common because the LUI is very powerful in the Linux system. The shell is typically used to start other programs.

The Terminal

In Linux, the terminal is actually a terminal emulator. The user is usually unable to communicate with the operating system without the use of the terminal, which is partially due to security purposes.

Note that the shell itself is a program that receives input from the terminal or GUI desktop - it is not the same thing as the terminal. Multiple terminals can be used to communicate with the same shell. However, although a terminal can talk to different variations of shells, it can only address 1 shell at a time.

The terminal is the screen that used to give the shell various commands. In Linux, sometimes only a terminal CLI is available because the machine does not need to run a desktop environment. This is okay for most computers and may even be necessary for those that have limited memory or do not need to have a screen the user can interact with.

Why Use a Command Line Interface?

For some smaller computers, such as what you may commonly find in robots or remote-controlled machines, user interaction with a desktop environment is impractical. In these cases, the machine is accessed by typing commands into the command line.

Even though it can seem inconvenient to those familiar with using a desktop, using the command line is faster, more powerful, and more practical. For space reasons, servers don't include a keyboard, mouse, and display so they must be accessed using the CLI. Small devices such as a home router may run a Linux OS and must be accessed with the CLI.

Now that you know a little about shells, terminals, and the command line, let's talk about the types of Linux shells and why you might want to use each of them.

Types of Linux Shells

In this chapter, we'll be talking about the two kinds of shells you may find in Linux: Bash and csh. Bash was originally used on Unix. The Apple OS was based on Unix and uses Bash so if you are familiar with the Apple OS command line, a lot of this may be familiar to you.

Mac and Windows Users

However, even if you are familiar with the Unix command line, there may be slight differences in Linux syntax.

If you are a Windows user, unfortunately, a lot of this may seem foreign to you. Even though Windows has the Windows Command Prompt, you may not be familiar with it if you have only been using your desktop environment to perform tasks.

Where To Run A Linux Command Line

There are different ways to get a Linux command line for learning.

Install Ubuntu

The best way is to install Ubuntu. There are a number of options for doing this including installing a dual boot Ubuntu image, running from a USB flashdrive (called USB Live), or installing Ubuntu on a VirtualBox image.

Dual Boot (Native Install)

Follow the Ubuntu tutorial to download the image to a USB flashdrive and install it.

- Pros – Choose whether to run Ubuntu or Windows at startup.
- Cons –Must reboot to switch to the other OS.

USB Live

Follow the Ubuntu tutorial to download the image to a USB flashdrive and but only run it from the USB drive.

- Pros – Does not change the hard drive in default.
- Cons – Harder to save anything. Changes don't persist – a reboot will go back to the initial state.

VirtualBox

Create a new VirtualBox environment and Ubuntu tutorial to download the image to your PC and install it.

- Pros – Can switch between different OSes easily.
- Cons – Uses space on hard drive. Slower than native install.

Webinal – Command Line in Browser

Another option is running the Command Line in a browser. Web options such as Webinal allow you to learn the Linux command line using a browser. In fact, a lot of the examples in this book use webinal.

Chrome OS and Others

If you already have a machine that runs an OS based off Linux, such as Google Chrome, you may fall into one of two camps - 1) you may have never even heard of the Crosh terminal and may have stuck to using your machine for light tasks, such as doing schoolwork, surfing the internet, or streaming movies and TV shows. In this case, you might have little to no knowledge of the Linux command line and little to no experience using a terminal.

On the other hand, if you've completed more advanced tasks with your Chromebook, you might have some familiarity with its inner workings and the Crosh terminal. In that case, some of this book may have been familiar to you, namely commands like sudo, apt-get install, among others.

That said, even though Google ChromeOS is based on the Linux operating system, they are not the same. If you are using dual OS mode to follow along with these exercises, make sure that you are not in Google Chrome and that you are not typing in a Crosh terminal. Switch to the Linux side to complete these exercises there to avoid any confusion.

Useful Commands

Tab-Completion

If you want to re-type a command but aren't sure what it looks like, you can start typing the command and press the **Tab** key. The terminal will attempt to complete the command for you, preventing you from having to stop to look up syntax and spelling for commonly used commands.

CTRL+R--Search History

To use this command, tap the **Ctrl+r** keyboard shortcut. You should get an option to search the history of commands you have typed into the terminal. Type in a few letters of what you're looking for and the last command you entered that contains those letters should show up.

You can also use the **Up Arrow key** to view the last commands you used to a point.

Other Shells

csh

The abbreviation csh stands for **C Shell** in Linux, or you may also get the version called **Tcsh** . The C shell is an LUI interpreter with a C programming language-like syntax. If you are not already familiar with the C programming language, you may want to familiarize yourself with it before you attempt to do any serious programming in a C shell.

Brief Overview of Shell Scripts

You can use either Bash or csh shells to run programs from a terminal. However, this book will cover Bash shell scripting.

The difference is the latter of the two will accept inputs using C-like scripting syntax. However, teaching C language is beyond the scope of this book, and may make learning Linux seem complicated. For now, we will stick to the Bash shell for Linux commands.

Chapter Summary

- csh and Bash are both types of shells used in Linux.
- csh, or C shell, is a type of shell that uses C-like syntax and scripting language.
- A terminal and a shell are not the same thing - a terminal is a program used to allow the user to give instructions to the shell.

CHAPTER FOUR: LINUX BASIC FILE DIRECTORY

Linux is modeled on the UNIX operating system. As such, the filesystem and commands are similar to UNIX. If you are already familiar with systems like Apple OS, the file system may be familiar to you.

It's also recommended that you use a bootable Linux USB stick for learning Linux. Changes are not stored on a bootable Linux USB stick so you can easily restart if you have a problem or make a mistake.

What is the Linux Filesystem?

The Linux filesystem is a collection of directories, and it is composed of various files and folders that keep it organized. You may be familiar with this concept if you've ever worked in an office. A "file system," whether we're talking about computers or not, is a system of organizing information.

In Linux, every piece of information is represented by a file. Everything looks like a file.

- The command that boots the operating system
- User passwords
- Actual text files
- The program to run removable hardware and flash drives.
- Devices such as a hard drive or UART output

It's all a file that a person can read, access, delete, and modify.

These files are stored in folders called directories, which are then stored in other directory folders, and so on. It's like having a folder full of files, stuffed into another larger folder, stuffed into another even larger folder and so on.

We will show a file hierarchy of folders as

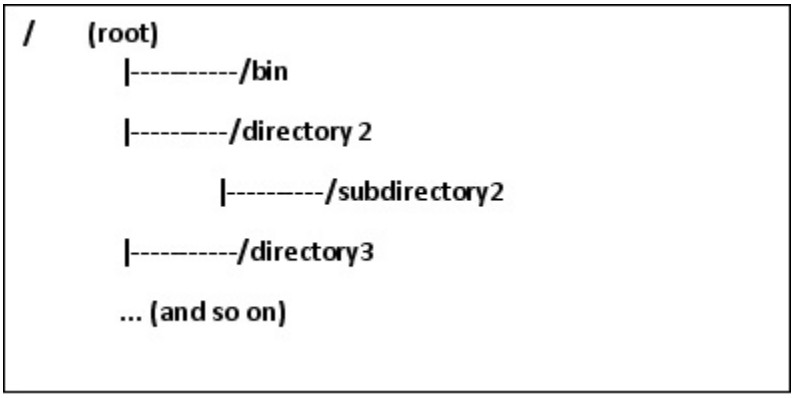
/var/log/mail

Where mail is a subfolder of log which is a subfolder of var which is a subfolder of root.

Visualizing the Linux Filesystem

In the Linux Command Line section, we'll learn about the tree command. For

now, we'll just show the output of the tree command.



The root directory has many sub-directories. Without these files and directories, certain programs and bits of hardware and software would not function at all on the computer. Because everything is a file in Linux, it's not uncommon for crucial parts of a program to be stored in separate folders.

It can also be difficult to find a file that you've downloaded if you don't know where to look.

Let's begin by looking at the Linux file system, what certain folder names represent, and where a downloaded file is likely to end up.

/	root (top level)
/bin	binaries (applications)
/boot	boot (startup)
/dev	device
/etc	system configuration
/home	home directory
/lib	libraries
/media	removable mount drives
/mnt	permanent mount drives
/opt	optional software
/proc	process information
/root	root home directory
/sbin	secure binaries

/sys	system status
/tmp	temporary directory
/usr	user data
/var	variable files (log)

Picture a House

Think of the Linux Filesystem as a house. Different rooms in the house have different functions. As we wander through the filesystem below, we'll also describe a house to make it easier to remember its function.

root

The / represents the "root" of the filesystem.

The forward slash is a symbol representing the "root" or "main" directory. The **main directory** should not contain any user files for optimum performance and to prevent clutter. But some versions of Linux, such as Debian, allow a root user to customize any parts of the program as he or she sees fit.

Ubuntu does not allow any single user automatic access to the / folder. Instead, it requires that the user enter special commands to access its contents. This will be covered in more detail later in this book, but that command is sudo, or "super user do." Even if you have administrator privileges, if you are using an Ubuntu-based system, you will have to enter sudo to execute commands that are only supposed to be carried out by the root user.

The main part of the house is the Living Room. All the other rooms branch off of the Living Room. The Linux Filesystem "Living Room" is the root directory. If you were to search the filesystem hierarchy, you wouldn't find a file named "root". The root folder is at the top in the picture above and is designated / . Like root, the homeowner has to open the front door and invite the visitor in.

bin

The letters bin stand for "binary." This folder contains the built-in executable programs that are normally executed.

The absolute path of this subdirectory is simply, /bin. As mentioned in the last section, the /, or forward slash, represents the root directory, while bin is a subdirectory inside of, or below, the root directory.

In our Linux House, this is the kitchen where the awesome food dishes are cooked up. These /bin programs are the ingredients that we can mix together to exciting dishes. We'll learn more about these built-in programs (ingredients) in the Linux Command Line chapters.

boot

This folder also contains boot-related files and it's best that the end-user not tamper with the contents of this folder unless that user knows what he or she is doing. In Linux, it is possible to put the operating system in a state where it will not function properly. You typically won't spend much time in this directory.

In our Linux House, this is the sidewalk leading up to the house. The boot directory leads you into the house just like the boot directory has the instructions for starting up and getting you into Linux.

dev

Think of the abbreviation dev as standing for "device files." This folder contains data for all of the essential devices installed on your system. Unlike Windows, Linux device files are accessible by the root user, or admin, depending on what permissions the user has. In some cases, the admin can even access the files controlling the hard drive and other essential equipment.

The devices need kernel drivers to connect to the Linux kernel. Some Linux kernel drivers also expose their interfaces to users through this directory. An example is the older UARTs that print out the logs in Raspberry Pi and other mini-computers.

In our Linux House, these are the appliances of the house. The stove, the refrigerator, the washing machine, and the sinks and toilets are all available for the family to use. The Linux kernel drivers are the electrical wiring and plumbing connecting the appliances to the outside world. The Linux drivers are hidden from view just like the wiring and plumbing are in the walls of the house. Some drivers expose their interfaces through the dev directory just like a sink exposes the water to the family.

etc

The etc folder contains configuration data for individual programs that are used system-wide. This is not the same as the individual user's etc folder. If the user has the proper permissions, they can hand-edit the data in these folders using a

text editor.

home

Individual user folders are found in the home folder. On the Ubuntu Desktop, the Download folder and Documents folder can be found under the home directory. On a command line, it might look something like, /home/username. For example, the command line would look like /home/kevin if the user's name happens to be Kevin.

The home directory has a shortcut symbol which is the ~. We'll discuss it more in the Linux Command Line sections but kevin can quickly jump to the home directory by typing either cd ~ or cd /home/kevin.

In our Linux House, these are the bedrooms of the house. Kevin has his own bedroom and Suzy has her own bedroom. They keep their private possessions in their separate bedrooms. The bedroom is similar to the home directory. Each user's personal setup is stored in the home directory. Each user can create and add files to their home directory as they see fit just like each person can decorate their bedroom.

lib

The Linux system library, which may look like /lib in the terminal as the direct path, contains libraries shared by application programs that are necessary for them to run. The libraries are needed for the application programs to run properly.

In our Linux House, this is the pantry with the spices that are used for cooking. The spices aren't used for every meal but they're available when they are needed. The main ingredients were the programs in the /bin directory and to taste just right, the spices in the pantry are needed.

media

The /media folder, or directory, contains subdirectories and subfolders where removable devices that store media can be mounted. Without this directory, devices such as USB drives and DVD ROMS might not work properly.

In our Linux House, this is the driveway to the house. Visitors to the house (USB) must park in the driveway before making their way to the house.

How to Use:

When you insert a USB drive or new removable device with media files, look for it to be added to this folder.

mnt

The /mnt is referred to as the "mount" command. This command enables you to mount and unmount files. The /mnt directory provides a place to mount more permanent files such as the CD or DVD drive.

In our Linux House, this is the garage attached to the house. Permanent residents of the house park in the garage instead of in the driveway.

opt

This directory holds software applications physically configured by the user or administrator. You may find files for your desktop environment and other software non-essential to the system here.

In our Linux House, this is the back patio. It provides some needed recreational space but it isn't an important part of the house.

proc

The /proc directory holds many details about your Linux system. The files stored here are referred to as "virtual" files because they don't actually exist on the disk. If you have the proper permissions, you can still explore this directory to learn more about the Linux system and programs on the computer you're running. It may help to picture the files in the proc directory as nothing more than ghosts of recently accessed files.

In our Linux House, this includes the thermostat, the water meter, and the electrical usage meter. If you want to see what is going on in your house, you'd check the thermostat to see what the temperature is, the water meter to see how much water you are using, and the electrical meter to see how much electricity you are currently using.

root

This is the home directory for the root user. Root is considered the highest level of anything, so the root user is the highest level user on the computer.

In our Linux House, this is the master bedroom. When you're in the living room, you're saying this is Jesse's house but when you're in Jesse's bedroom, that is where his personal things are kept.

sbin

The /sbin directory holds system commands that affect the entire system, and only the root user is allowed to run them.

In our Linux House, this is the private library or home office that includes the plans for the house, the house insurance, and the builder's warranty on the house. The root occupant is usually the only one that uses this room.

sys

The /sys directory stores temporary files with device information in a virtual file system. This is very similar to the /proc directory and is basically a newer and better-maintained version of the /proc directory. The /sys directory typically has more ways to control device drivers while the /proc was typically for reading.

In our Linux House, this includes the thermostat and meters as well as the light switches, faucets and circuit breakers. We can turn the water and electricity off and on as desired.

tmp

This directory stores temporary files from applications and users. Do not store anything important in the /tmp folder, as the contents may be regularly deleted when a user logs off or the system is shut down.

In our Linux House, this is the trash can.

usr

Although it looks like an abbreviation for "user," it's not. It stands for "Universal System Resources" or, on Unix, "Unix System Resources." Regardless, the /usr directory still stores much of the information used by individual user accounts on a Linux system, including binaries, header files, and documentation for the binaries.

In our Linux House, this is the drawer that has all the user manuals and warranty information for the appliances.

var

The /var folder contains files that the system writes data to while it's operating. var is a high-level directory that is not shared over a network or with other computers, and it contains some user information that is not read-only. This is

the folder where the log files that report on system problems or health are stored.

In our Linux House, this is the laundry room. When you walk into the laundry room, you can determine from the clothes that Kevin was playing in the mud at the playground and Suzy had an ice cream cone that melted before she could finish it.

Common Shortcut Paths

. A single period means the current directory

.. Two periods means the parent directory so the directory above the current directory

~ The user's home directory. Each user has their own home directory (private bedroom).

/ The root directory.

Relative and Absolute Paths

The relative path of a file is the path of that file from the current working directory. For example, if you were currently working in the home file and you stored something that keeps track of user passwords, you would be able to access that file simply by typing the filename, and maybe sudo if your account does not have administrator permission.

In our house example, we can describe a relative path as

On the other hand, if you are working in another folder or directory, let's say, the root directory, and you typed in just the name of the file, you might come up with an error. Remember that, in the previous scenario, you stored the file in the home directory and not the root directory. Merely typing in the name won't work--you will either have to change directories and then type in the relative path.

Another option is to type in the absolute path which is the entire path to that file starting at the root directory. Let's assume that you already know the path, in this case, let's say it's something like, /home/usr/passwords. Typing this whole path into the terminal would be one way to get to the file you need.

Quiz Time:

Take out your notebook and answer the following questions. Feel free to look at an earlier part of the chapter if you think you need a reminder.

What is housed inside the Linux file system?

Why is it important to know what kind of files each folder is designated to handle?

Answers:

The Linux file system houses files and directories that are vital to the operation of the system in addition to files that are important to the user.

Linux has designations for what certain programs do, based on where they are stored.

Where you store a file in Linux is important, but it's also important to understand syntax. You might have noticed that the cd and ls commands follow a certain format. What did they have in common? For one thing, they were all lowercase. Capitalization or lack thereof is important in Linux, which is a concept we will go over in the next section.

Capitalization and Syntax in Linux

Capitalization does not necessarily matter in Windows, but in Linux, lowercase and capital letters are treated as different letters. For example, a folder called "home" would not be the same as "Home." It is also true of your username login.

Did you notice that some of the subheadings in this book are not capitalized? It's not an oversight by the editor - in Linux, capitalization, or lack thereof, matters, and those subheadings were written to remind you how to properly write the commands.

Linux is very sensitive when it comes to capitalization and the correct usage of syntax. You may want to double-check the commands you type to make sure that you've typed in the correct syntax. A missing period, misplaced colon, or any other identifying sign could result in an error or could lead you to a new directory entirely.

Tips for Completing Exercises in Future Chapters:

To avoid any issues, it's highly recommended that you look at all examples very carefully and make sure you understand what is being presented before practicing on your own. If there is ever any doubt and you have internet access, you should consult the internet, especially the Linux forums and tutorial websites, for more information. However, it is best to search for the answer yourself if you want to strengthen your programming abilities and understanding

of the Linux system.

Let's say you type in a directory path and get an error. Look at the picture and provided examples carefully--did the instructions tell you to type in a backward slash or a forward slash? Did you miss any steps, such as first checking if that directory existed?

If you don't understand the information presented, re-read it while carefully taking notes. Ask yourself, "what exactly do I not understand?" If it's something simple, for example, you aren't sure whether the command is asking for a capital "I" or a lowercase "l," try applying some logic. What does the command stand for? If it appears to stand for something that starts with an "l," such as list, it's reasonable to guess that it's a lowercase "l."

You should also keep in mind that, with a few exceptions, most commands in Linux are in lowercase. If there's a time where they are not in lowercase, it is one of those few exceptions and will be clear from the context. If there is ever any doubt go with lowercase.

Although this book is designed to be beginner-friendly, it may appear confusing at first while you are familiarizing yourself with the Linux terminal and trying to learn the commands. Don't get frustrated if you type a command and receive an error message. You may have mistyped or entered the wrong letter or symbol, which is easy to do if you are just starting.

There's also a good chance that you're simply in the wrong directory or you may be attempting to perform operations that you don't have permission to perform, depending on what version of Linux you're using. These errors appear more often if you are searching for online help without a full understanding of your system. This should be less of a problem for you if you're using the latest version of Ubuntu Linux because the examples in this book assume you are using Ubuntu Linux.

If you still want to follow along using another distribution of Linux out of convenience for your situation, feel free to do so. You can try learning Kali Linux or Fedora for server maintenance and you don't want to go through the extra hassle of learning another version of Linux, for example.

You should be able to pick up the syntax of your distro of choice fairly easily - just don't get discouraged if the commands don't work exactly the same in different versions of Linux. The syntax may differ but the basic commands and functions are the same; you may just have to learn slight variations in syntax to get them to work for you.

If you would like more information about the Linux file hierarchy, there is a lot of useful information about it online. Feel free to do a Google search if you are interested.

Chapter Summary

- The Linux file system is like a hierarchy, with the top files being the main parts of the directory and smaller, sub-directories underneath.
- Linux stores everything as a file
- Certain file directories complete certain tasks to avoid clutter or confusion.
- The /proc directory can tell you a lot of information about the particular system.

CHAPTER FIVE: THE LINUX COMMAND LINE - THE BASICS

Everything in Linux, from your root filesystem to your programs, is stored as a file or directory. In fact, even a directory is a file. To gain full access to the files in Linux, you may need to learn how to use the command line. The command line is a more powerful way to access files than the GUI and often gives the user more options than using a GUI interface.

To review, a GUI (Graphical User Interface) is the screen that is displayed that contains clickable icons to access the programming files you need. Most everyday functions can be carried out using the GUI in most versions of Linux. However, to become a Linux power-user, you must learn to use the command line for more flexibility and control over the filesystem and programs. The command line is sometimes called the **line user interface** , or **LUI** . It is most commonly referred to as the **command line interface** or **CLI** . In this chapter, we will be covering some basic commands to help you get more comfortable with using the CLI.

There are multiple ways to do these exercises on a computer. You can run a complete Ubuntu installation in VirtualBox as covered in Appendix A. Ubuntu can also be run from a bootable Live USB image but you won't be able to jump back and forth between Ubuntu and Windows/Mac like you can with the VirtualBox installation.

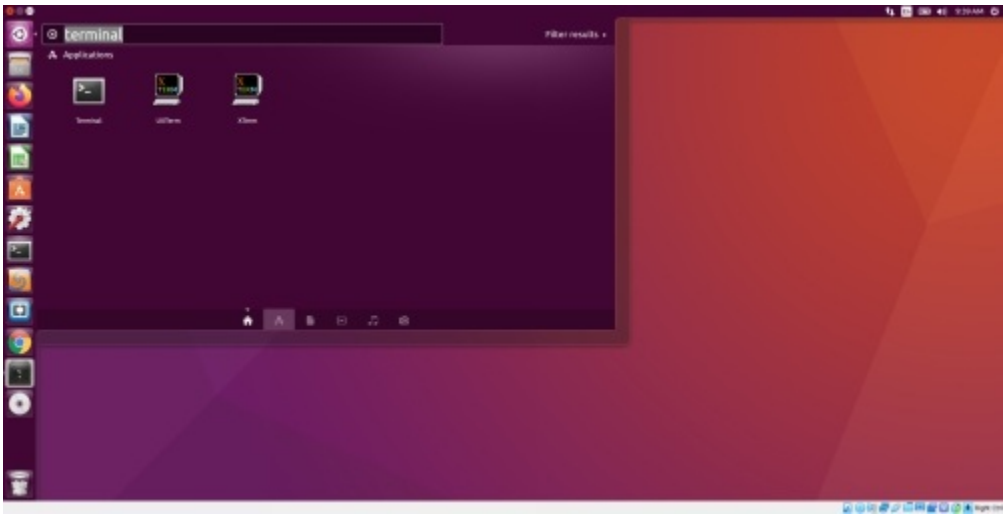
Another option is opening up a virtual terminal emulator in a browser. To find terminal emulators, try searching for "virtual Linux terminal online". I actually like using webinal.org because the terminal starts at a user-mode prompt (~) instead of a root user mode prompt (# - explained in more detail below).

One good resource with a list of virtual Linux terminals was https://itsfoss.com/online-linux-terminals/

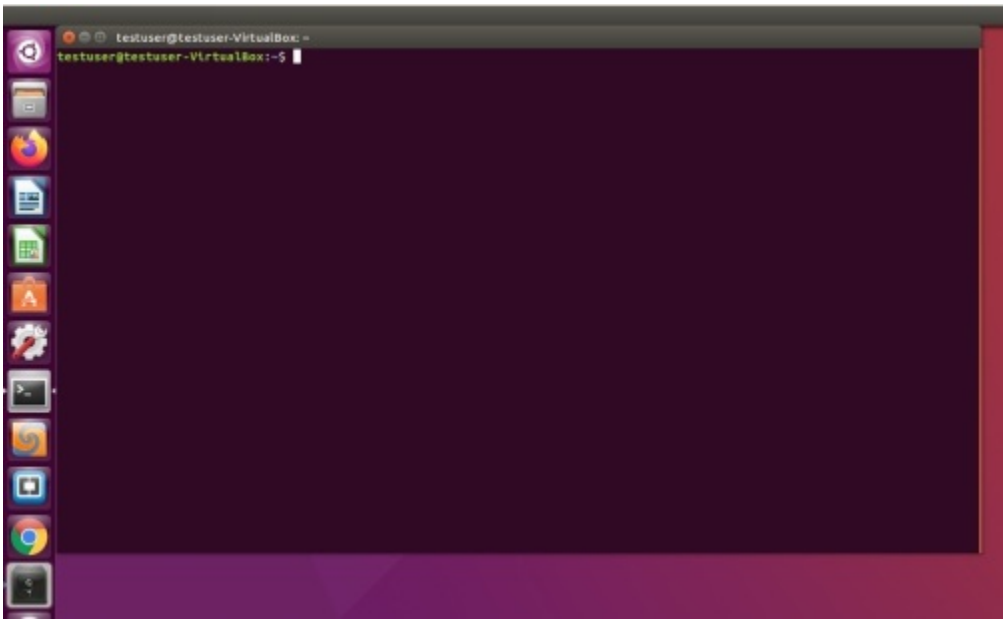
Before we begin, this would be a good time to grab a pencil and a small notebook to take notes. You are more likely to learn by actually doing the exercises and taking careful notes. Some of the commands listed can seem somewhat similar if you are just starting out. Recognize that memorizing all of these commands can take time, so slow down and work smart. Take notes to prevent yourself from feeling overwhelmed, as the human mind can only absorb a little bit of information at a time. To successfully learn something, you should break the information up into small chunks and not try to cram it in all at once.

Alright, with our pencil and notepad ready and the computer on, let's get started.

We will be using Ubuntu in our examples. To get started, open up a terminal. As shown below, you will search for Terminal and then click on it.



The Terminal application should look like this:



Open up a terminal - you should see something like username@machinename~$.

The dollar sign, $, is used to denote user mode. It differs from the pound sign or hashtag #, which is used to denote root user mode. Remember that you do not get access to the root user directly in Ubuntu; you have to type in sudo, which is the command to access the root directory, and then your command. We'll talk

about how to do that in another chapter. For now, let's focus on some basic commands.

The Command Line - Basic Commands

Command	Description
pwd	**Print Working Directory**
cd	**Change Directory**
clear	**Clear the terminal**
ls	**List Directory**
file	**File characteristics**
mv	**Move a file or directory**
cp	**Copy a file**
mkdir	**Make Directory**
rm	**Remove a file**
rmdir	**Remove an empty directory**
touch	**Create an empty file**
cat	**Concatenate a file (dump contents of result)**
echo	**Echo the contents to the terminal**
less	**Display the contents of a file**
head	**Display the first 20 lines of a file**
tail	**Display the last 20 lines of a file**

Command Options:

Sometimes you can add options to commands to make them do different things. For example, you could add -b to your file command to display a brief version. This may come in handy if all you want to know is the file type in brief mode.

When combining this command with options, use the syntax: file [option] [filename]. For a complete list of options, see the man command, or manual pages by typing man after the command (ie file man) into the terminal. You may also be able to use the --help option such as ls --help

#

Basic Navigation

The following commands are used to move around in the filesystem.

#

pwd

pwd stands for "print working directory," and when this command is used, it will print the full path of your present working directory on the screen. To use this, just type pwd and press **Enter** . You should get something like, /users/files/namedfolder.

Example:

```
[linux12@webminal.org~]$ pwd

/home/linux12

[linux12@webminal.org~]$
```

In the example above, the present working directory is /home/linux12 because we're logged in as the user linux12. We could have guessed this from the ~ in the bracket path. We can also see that we're in user mode because of the $ at the prompt.

Explanation:

Linux systems may require you to enter very specific commands to receive the results you want. The folder you are working in matters when determining whether or not these commands are successful. A program isn't going to assume you have access to anything other than your current folder. But sometimes you may not be sure where you are and the pwd command lets you know where you are. In this case, we're in one of the bedrooms of the Linux house.

#

cd

This command stands for "change directory." To change to a lower level directory than the one you're already working in, type cd, press the **Spacebar** and then type the name of the directory to move to. Let's say we're trying to access a folder called "my_test" in the current directory. We would type cd my_test, and then we would press **Enter** .

Doing so is a request to find "my_test" in the current folder. Those not familiar with navigating files and folders in Linux may think that relative commands do not work and the only reliable way to access files and folders is by typing in the absolute path, such as cd misc/var/web. While Linux does recognize relative commands, there is a slightly different syntax.

Example

```
[linux12@webminal.org~]$ cd my_test/
[linux12@webminal.org my_test]$pwd
/home/linux12/my_test
[linux12@webminal.org my_test]$ cd ..
[linux12@webminal.org~]$ cd /
[linux12@webminal.org/]$ pwd
/
[linux12@webminal.org/]$ cd-
/home/linux12
[linux12@webminal.org~]$
```

Explanation:

A lot is going on in the example above. We move into the my_test directory and print the working directory to make sure we are where we think we are. The ".." means the parent directory so the "cd .." takes us back to the home directory where we came from. We then change into the root directory and confirm our location. The - is used to take us back to the last directory we were in so the "cd - " takes us back to the home directory.

A lot of tutorials on the internet about using the Linux command line may make the process appear more complicated than it really is. A person new to using the command line and unfamiliar with using relative commands may make the mistake of typing out the full path every time he or she tries to execute a command. You don't have to do this.

Remember that Linux uses a different kind of syntax than Windows. For example, typing / before the folder name is like asking it to find a folder in root. If the folder is not in root, you may receive an error message. Likewise, if you try to find a folder in ~, or home, and the folder is not in that directory, you may

also receive an error message.

Use the pwd and the cd commands to move around and understand the filesystem structure.

Example:

Let's go from your home, or ~, folder to your Desktop folder on the command line. Just type cd Desktop into the command line and press **Enter** . You should now see something like this:

```
[linux12@webminal.org~]$ cd Desktop
[linux12@webminal.org Desktop]$ pwd
/home/linux12/Desktop
[linux12@webminal.org Desktop]$
```

To go back, type cd $HOME, or cd ~ ($HOME and ~ represent the same home directory in this context). Both of these will take you back to your home folder. You should now see something like below:

```
[linux12@webminal.org Desktop]$ cd ~
[linux12@webminal.org~]$ pwd
/home/linux12
[linux12@webminal.org~]$
```

Do you notice how the directory is different from when you were in your Desktop folder?

Exercise:

Open the command line on your Linux device or virtual machine and type, ls. This should bring up a list of files and folders on your device. Next, use the "change directory" command. Type, cd, press the **Spacebar** , then type and the name of a folder you would like access to. For this exercise, let's use, etc. To ensure you've gone to the correct directory, use the pwd command, which will be explained below.

#

Files Types & Organization

When using the command line, you may want to check the specifics of a file. Is it a file or a directory? Is it a text file or an executable?

#

clear

When you are typing commands into the terminal, it can often become quite cluttered, especially if you've recently downloaded new software using the command line. When the screen gets cluttered, it can make it harder to focus on your next command. The cluttered commands can also get in the way of commands you want to implement later. Although using **Escape** , **Delete** , and **Backspace** will not work to remove your old commands, you don't necessarily have to close the terminal and open a new one if all you want is to clear the screen.

Just type, clear into the terminal after the $. The screen should clear itself, and you should be able to resume typing commands onto the screen.

#

ls

You can use the ls, or "list" command, to display files in the current working directory. Remember that the current working directory is the directory on your current command line that you are working with. As you move into a new directory, that becomes your current working directory.

Type ls to have a list of files generated. The ls by itself command lists all the files in the current directory. However, if you put a path after the ls, you can list the files in any directory.

Type ls / to see the files in the root directory or ls /etc/systemd to see the files in the /etc/systemd directory

Example:

```
[linux12@webminal.org~]$ ls

Desktop my_test textfile.txt

[linux12@webminal.org~]$ ls /

bin common_pool etc lib lost+found mnt proc run srv test tmp usr

boot dev home lib64 media opt root sbin sys test2 tmpuserdata
var

[linux12@webminal.org~]$ ls /etc/systemd

ls: cannot access /etc/systemd/system: Permission denied

bootchart.conf journald.conf system user

coredump.conf logind.conf system.conf user.conf

[linux12@webminal.org~]$
```

Helpful Options

The output of the ls command isn't very helpful. 3 values are printed out on the same line. There are 2 of these values in bold (Desktop and my_test) to signify a directory. Many options can be used to tailor the ls command to your needs. We will only cover a few of the most used options but you can see the entire list by typing ls --help (there are a lot of options so I won't capture them here).

-l : long listing format (this is a lower-case l and not a #1)

-a : list all the files including hidden files (preceded with a .)

-s : Sort by size

A better way to display the current directory is with ls -l because it gives much more useful information.

Example:

```
[linux12@webminal.org~]$ ls -l
total 0
drwxrwxr-x. 2 linux12 linux12 6 Feb  2 19:37 Desktop
drwxrwxr-x. 2 linux12 linux12 6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 18:29 textfile.txt
[linux12@webminal.org~]$
```

Explanation

See how this is much easier to read with each file on a different line? Notice the "d" in the left-hand column of Desktop and my_test is used to indicate a directory. Also note the modification timestamp of the file - which was February 2 at around 6pm.

We will discuss the **rwx** fields at a later time.

To see the hidden files in a directory, type ls -al to show everything in the directory in long format.

Example:

```
[linux12@webminal.org~]$ ls -al
total 5932
drwx------.   4 linux12 linux12    4096 Feb  2 19:37 .
drwxr-xr-x. 171965 root   root    4083712 Feb  2 19:35 ..
-rw-r--r--.   1 linux12 linux12       0 May 31  2017 .bash_history
-rw-r--r--.   1 linux12 linux12      18 Dec  7  2016 .bash_logout
-rw-r--r--.   1 linux12 linux12     193 Dec  7  2016 .bash_profile
-rw-r--r--.   1 linux12 linux12     231 Dec  7  2016 .bashrc
drwxrwxr-x.   2 linux12 linux12       6 Feb  2 19:37 Desktop
-rw-r--r--.   1 linux12 linux12     334 Sep 20  2017 .emacs
-rw-r--r--.   1 root   root         9 Feb  2 17:38 .magic_string.txt
drwxrwxr-x.   2 linux12 linux12       6 Feb  2 18:28 my_test
-rw-rw-r--.   1 linux12 linux12       0 Feb  2 18:29 textfile.txt
-rw-r--r--.   1 linux12 linux12     658 Aug  2  2017 .zshrc
[linux12@webminal.org~]$
```

Explanation

Notice the files with the “.” preceding the name. This indicates a hidden file. At the end of the Linux Basic File Directory chapter, we discussed the . which indicates the current directory and the .. which indicates the parent directory and are shown at the top of the dump. They both have a d indicating they are directories.

#

file

Use this command to have your computer tell you a file's type. All you have to do is enter the file followed by the name of the file of which you want to know the type. The file command can be helpful when you aren't sure how to handle a specific file. Let's say you've saved several different kinds of files under the same name (my_name.png, my_name.txt, and my_name.pdf), or you need to help rewrite a specific file on the system, but you aren't sure what kind of file it

is. You can use the file command to help you figure that out.

For example, if I wanted to know what type of file "file textfile.txt" was or even what the cd command was, I would need to type the following into the terminal.

Example:

```
[linux12@webminal.org~]$ file textfile.txt
textfile.txt: ASCII text
[linux12@webminal.org~]$ file /bin/cd
/bin/cd: POSIX shell script, ASCII text executable
[linux12@webminal.org~]$ file *
Desktop:    directory
my_test:    directory
textfile.txt: ASCII text
[linux12@webminal.org~]$
```

Explanation:

The file command tells us the textfile.txt file has ASCII text which is just text like you are reading. However, the cd command used to change directories is a shell script made from ASCII text which is executable. As discussed in the Linux Basic File Directory chapter, the /bin directory is the kitchen and holds the common Linux command line programs we'll be running.

You can also use this command to find all of the files in your current working directory by typing file *. The asterisk, *, signals to the machine that you want it to show you the file characteristics of everything that Linux considers a file in the current directory.

#

Manipulating Files in Linux

These commands are used to manipulate files including create a directory, move, copy, and delete.

#

mv

If you type mv into the terminal, you are telling Linux to move a file. To use this command, type, mv and the name of the file you want to move, then type the name of the location where you want to move that file. Once you've done all that, hit **Enter** . The mv command is also used to rename a file - think of this as moving the file to a new name.

Example:

Let's say you want to move a file called "photos.png" to my home Desktop directory. All you would need to do is type the following into the terminal:

```
[linux12@webminal.org~]$ mv photo.png ~/Desktop/
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 2 linux12 linux12 22 Feb  2 21:17 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$ ls -l ./Desktop/
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:10 photo.png
[linux12@webminal.org~]$
```

If you do everything correctly, the file "photos.png" would appear in my Desktop directory. The Desktop directory is under the home directory /home/<username> because ~ refers to the home directory.

To rename the file, you can go to the "Desktop" directory and type in the new filename, mv photos.png photos. It should now appear in your Desktop folder as a photos file. Notice that it doesn't have the png at the end so we would need to use the file photos to figure out that it was a .png file.

```
[linux12@webminal.org~]$ pwd
/home/linux12
[linux12@webminal.org~]$ cd Desktop/
[linux12@webminal.org Desktop]$ ls -l
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:10 photo.png
[linux12@webminal.org Desktop]$ mv photo.png photos2.png
[linux12@webminal.org Desktop]$ ls -l
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:10 photos2.png
[linux12@webminal.org Desktop]$
```

#

cp

When you type cp into the terminal, it stands for "copy file." In a sense, this command is used similarly to the mv command. To make a copy of a file with a new filename in the same directory, you would simply need to type cp and the name of the current file, then the name of the new file you want to copy to.

When copying directories, you may receive a message saying you need to put an -r after cp to copy the files or directories. The -r is for recursive and indicates you want to copy the directory and all the contents inside the directory.

Example:

First, we'll figure out where we are. Use the **pwd** to find out where we are.

```
[linux12@webminal.org~]$ pwd
/home/linux12
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:21 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

This shows the current directory.

Now, let's copy the photos2.png file from Desktop/photos to the current directory and rename it photos.png.

```
[linux12@webminal.org~]$ cp Desktop/photos2.png ./photos.png
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:21 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

But what about copying a complete directory? Can we just copy a directory the same way?

```
[linux12@webminal.org~]$ cp Desktop Desktop2
cp: omitting directory 'Desktop'
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:21 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

Nope. That failed because the directory has files in it. If the directory were empty, the copy would succeed.

To copy the directory and all the files inside it, we need to use the -r option which does a recursive copy.

```
[linux12@webminal.org~]$ cp -r Desktop Desktop2
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:21 Desktop
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:34 Desktop2
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$ ls -l Desktop2
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:34 photos2.png
[linux12@webminal.org~]$
```

#

mkdir

The abbreviation mkdir stands for "make directory." Think of it kind of like the "create new folder" option in Windows. Entering this command in Linux will make a new directory where you specify. For example, if you type mkdir apples on the command line, it will create a new directory labeled "apples" in the directory where you are working.

You can also create a new directory and put it in a specific directory by specifying a path. For example, let's say you want to create a new folder called "photos" on your Desktop. At the command line, you would type mkdir ~/Desktop/photos, and a folder called "photos" would appear on your desktop.

Example:

```
[linux12@webminal.org~]$ ls -l Desktop
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:42 photos2.png
[linux12@webminal.org~]$ mkdir Desktop/my_photos
[linux12@webminal.org~]$ ls -l Desktop
total 0
drwxrwxr-x. 2 linux12 linux12 6 Feb  2 21:43 my_photos
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:42 photos2.png
[linux12@webminal.org~]$
```

We look at the Desktop directory and then we created a new directory in the Desktop directory. We confirmed the new directory with the **ls** command.

Now let’s try moving a file into the new directory

```
[linux12@webminal.org~]$ mv Desktop/photos.png Desktop/my_photos/
[linux12@webminal.org~]$ ls -l Desktop/my_photos/
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:42 photos.png
[linux12@webminal.org~]$
```

#

rm

If you want to remove a file, type rm and the name of the file you want to remove. Make sure the file you want to remove is in the directory you are currently working in, or it may return an error.

```
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:21 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$ cp photos.png photos_t.png
[linux12@webminal.org~]$
```

We've now made a copy of a file that we can now delete.

```
[linux12@webminal.org~]$ rm photos_t.png
[linux12@webminal.org~]$
```

Removing a Full Directory

If you want to delete a directory that has items in it, you may have to add an -r after the rm command to indicate the remove is recursive. When you do this, it will look something like rm -r dir1 instead of just rm dir1.

```
[linux12@webminal.org~]$ ls -l Desktop
total 0
drwxrwxr-x. 2 linux12 linux12 23 Feb  2 21:43 my_photos
[linux12@webminal.org~]$ cp -r Desktop Desktop3
[linux12@webminal.org~]$ ls -l Desktop3
total 0
drwxrwxr-x. 2 linux12 linux12 23 Feb  2 21:51 my_photos
[linux12@webminal.org~]$
```

We created a new directory named Desktop3. Why? So that we can delete it, of

course.

```
[linux12@webminal.org~]$ rm Desktop3
rmdir: failed to remove 'Desktop3': Directory not empty
[linux12@webminal.org~]$
```

But this acts the same as the copy command. The directory can't be deleted if it still has files in it. Remember how we copied the entire directory and files last time?

Yes, we need to use the -r command to recursively delete the files inside the directory as well as the my_photos directory.

```
[linux12@webminal.org~]$ rm -r Desktop3/
[linux12@webminal.org~]$ ls -l Desktop3/
ls: cannot access Desktop3/: No such file or directory
[linux12@webminal.org~]$
```

Now let's confirm the Desktop3 directory isn't there anymore.

```
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 3 linux12 linux12 22 Feb  2 21:50 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

The removal appears complete.

#

rmdir

This stands for "remove directory," and it looks like, rmdir on the command line. When you type rmdir and the name of the directory you want to remove, the directory should disappear. The directory has to be empty, however. This will not

work on a directory that already has something in it.

Removing a Full Directory

If you want to delete a directory that has items in it, you may have to add an -r after the rm command to indicate the remove is recursive. When you do this, it will look something like rm -r dir1 instead of just rm dir1.

```
[linux12@webminal.org~]$ ls -l Desktop
total 0
drwxrwxr-x. 2 linux12 linux12 23 Feb  2 21:43 my_photos
[linux12@webminal.org~]$ cp -r Desktop Desktop3
[linux12@webminal.org~]$ ls -l Desktop3
total 0
drwxrwxr-x. 2 linux12 linux12 23 Feb  2 21:51 my_photos
[linux12@webminal.org~]$ ls -l Desktop3/my_photos/
total 0
-rw-rw-r--. 1 linux12 linux12 0 Feb  2 21:51 photos.png
[linux12@webminal.org~]$
```

We created a new directory named Desktop3. Why? So that we can delete it, of course.

```
[linux12@webminal.org~]$ rmdir Desktop3/my_photos/
rmdir: failed to remove 'Desktop3/my_photos/': Directory not empty
[linux12@webminal.org~]$
```

```
[linux12@webminal.org~]$ rm -r Desktop3/my_photos/
[linux12@webminal.org~]$ ls -l Desktop3/my_photos/
ls: cannot access Desktop3/my_photos/: No such file or directory
[linux12@webminal.org~]$ ls -l Desktop3
total 0
[linux12@webminal.org~]$
```

But this acts the same as the rm command. The directory can't be deleted if it still has files in it. Remember how we deleted the entire directory and files last time? Yes, we need to use the -r command to recursively delete the files inside the directory as well as the my_photos directory.

Now that the directory is empty, we can use the rmdir command.

```
[linux12@webminal.org~]$ rmdir Desktop3/
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 3 linux12 linux12 22 Feb  2 21:50 Desktop
drwxrwxr-x. 2 linux12 linux12 19 Feb  2 21:34 Desktop2
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

Sure enough, the Desktop3 directory is now gone after running the **rmdir** command.

The **rmdir** command may come in handy if you create a file in a directory that you no longer need and it's inconvenient or not possible to remove the file using the desktop. As mentioned earlier, not all Linux systems have desktop

environments installed, so you may find yourself using this command frequently if you work somewhere where you need to create files and then delete them.

Be careful with the rm or rmdir commands because using them carelessly can damage your system and you could end up with missing files. To prevent costly errors, make sure to back up your data before removing files or directories at the command line.

#

Creating and Viewing Files

Being able to manipulate files may be a step in the right direction, but how do you create and view files?

touch

The touch command creates a blank file. Just type **touch** and the name of the file you would like created.

Example:

First, let's get a feel for what the directory looks like.

```
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 3 linux12 linux12 22 Feb  2 21:50 Desktop
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

We know what the current directory looks like. How can we create a new empty file?

```
[linux12@webminal.org~]$ touch mynewfile.txt
[linux12@webminal.org~]$ ls -l
total 4
drwxrwxr-x. 3 linux12 linux12 22 Feb  2 21:50 Desktop
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 22:00 mynewfile.txt
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

As you can see, the **touch** command created a new file with a size of 0 (see the 0 right before the Feb 2?)

Notice that the total didn't change (still 4) because total gives the number of blocks used. Since the file is 0 in size, nothing needs to change.

Exercise:

Create two empty text files using the touch command. Place them on your Desktop.

#

echo

Use the echo command when you want to echo a string of text. Just type echo and write the message you want to say in quotation marks. For example, echo, "hello world" would print out "hello world" to the screen on the terminal. This may come in handy if you are programming and want to leave personalized instructions to whoever uses the terminal next.

Other Uses for echo

You can also use **echo** like the **list** command. To do so, you would type "echo" and an asterisk. It should give you a list of all of the files and directories in your current working directory.

To put a string in a file, you can use the Linux redirect command which is the > symbol. The redirect symbol redirects the output into a file so you won't see it on

the screen. Let's give it a try:

Example:

Let’s try using the **echo** command.

```
[linux12@webminal.org~]$ echo "Hi Linux CLI experimenters"
Hi Linux CLI experimenters
[linux12@webminal.org~]$
```

Well, that doesn’t do much, does it? It just printed out a string – just like if we’d typed it.

Can we do something useful with it? How about creating a file with a greeting inside?

```
[linux12@webminal.org~]$ echo "Hi Linux CLI experimenters" > hello.txt
[linux12@webminal.org~]$ ls -l
total 8
drwxrwxr-x. 3 linux12 linux12 22 Feb  2 21:50 Desktop
-rw-rw-r--. 1 linux12 linux12 27 Feb  2 22:08 hello.txt
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 22:00 mynewfile.txt
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$
```

The hello.txt file has “Hi Linux CLI experimenter” in it – just as if you’d typed it into Notepad or a text editor.

Exercise:

Use the echo command to print "hello world" to the screen.

Pair the echo command with the * command and print a list of your current files and directories in your home folder. Record your findings in your notebook.

Using the options you know of now, create a new command using the echo

[options] [strings] syntax and enter it into your terminal and describe what it does. If it doesn't do anything, write that down, too. Experiment with adding two or three different commands and record your findings.

#

cat

To concatenate means to link multiple things together. You can combine multiple files by entering the **cat** command. A nice side-effect of the **cat** command is that it prints out the results of the concatenation operation on the screen. Just type, **cat** and the name of the file to view the contents of the file in the terminal.

You can do quite a bit with this command, including creating new files, viewing the contents of multiple files, displaying the output of a file, copying the contents of one file to another file, and appending the contents of one file to another.

Creating an Empty File

You can also use this command to create new empty files by typing, *cat > file1* . Doing so will create a file called "file1."

View the Contents of Files

For example, typing *cat file1* and then pressing **Enter** would allow you to see the contents of file1. To view more than one file with the **cat** command, just enter the command **cat** and the names of the files you want to be displayed with spaces in between their names.

For example, *cat file9 file10* would display the contents of file9 and file10 to the terminal emulator screen.

```
[linux12@webminal.org~]$ ls -l
total 8
drwxrwxr-x. 3 linux12 linux12 22 Feb  2 21:50 Desktop
-rw-rw-r--. 1 linux12 linux12 27 Feb  2 22:08 hello.txt
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 22:00 mynewfile.txt
drwxrwxr-x. 2 linux12 linux12  6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12  0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12 31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$ cat hello.txt
Hi Linux CLI experimenters
[linux12@webminal.org~]$
```

So we use the **cat** command to see the contents of the hello.txt file.

To concatenate means to link multiple things together. Think of **cat** as a command that you use to merge files together.

```
[linux12@webminal.org~]$ cat hello.txt  hello.txt hello.txt
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
[linux12@webminal.org~]$ cat hello.txt
Hi Linux CLI experimenters
[linux12@webminal.org~]$
```

We sent the contents of the hello.txt file to the output display 3 times. Just to verify that hello.txt didn't change, we **cat** it by itself again.

But if we send the resulting output to a file, **cat** is an even more useful command. To do this, we use the > symbol along with the name of the file. The > *filename* means send the output into a file instead of displaying it on the screen.

Let’s give it a try. We use the same command as above but this time, the output will be re-directed into a file.

```
[linux12@webminal.org~]$ cat hello.txt hello.txt hello.txt > hello3.txt
[linux12@webminal.org~]$ cat hello.txt
Hi Linux CLI experimenters
[linux12@webminal.org~]$ cat hello3.txt
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
[linux12@webminal.org~]$
```

Notice that there is no output on the screen when it is redirected to the file. I **cat** the original file - it is the same and then I **cat** hello3.txt and see the 3 copies of the string.

Exercise:

With this exercise, we will practice using the cat command to create new files at the command line, read those files, and merge them into one new document.

To complete this exercise, you may need a Linux operating system with a desktop environment installed.

Change directories to your desktop environment and use the echo and cat commands to create a new file. Create another new file with the cat command. Write whatever you want in this file as well.

Now cat both files so you can see their contents on the terminal. Once you've done that, use the cat command again to create a new file by merging the contents of the old files and redirecting the output to the new file.

Record your findings in your notebook after you complete the exercise. Use the rm command to get rid of the files you created if you have no other use for them.

#

less

The **less** command allows you to view the content of the files displayed on the screen in small chunks rather than as a giant wall of text that you may have trouble reading.

The **less** command stops at each page and you can use the up/down arrows and the page up and page down buttons to get to the end. This command helps you quickly scan files for important information while avoiding wasting time wading through the information that you don't need.

You can also type a search term after “/” and have every occurrence of the term you are searching for highlighted, making it much easier to find the specific information you're looking for. It is similar to the **Ctrl+f** keyboard shortcut Windows, Chrome OS, and Mac.

To exit less, type the letter "q" (which means quit).

Exercise:

Go online and copy some Lorem Ipsum text or write a long document. View its contents using the **cat** command and notice how it scrolls past very fast. Then try using **less** to view it.

Also, try adding a / and using the search option. Record your findings in your notebook.

#

head

If you want to see the beginning of a specific file, the **head** command is what you're looking for. Typing in **head** and the name of the file that you want to be read will result in the first ten lines of the file's content being printed out onto the screen of the terminal.

The head command shows the first ten lines of the contents of a specific file by default, and you can adjust it using the -n option and specifying a number. For example, let's pretend that you have a text file with over 700,000 lines of code, but you only need the first 17 lines. You can use the head command with -n 17 and the filename to retrieve that information without having to open the entire file with all 700,000 lines of code.

The **head** command helps reduce clutter if you decide to read a large file and you already know that the information you need is at the beginning of the file. For instance, the title of a document is probably one of the first lines in the file

along with the date. If you only need that information, the head command is faster and cleaner than using the **less** command.

It also might help the terminal use less memory printing and processing text while saving you time when looking through large files.

Example:

By default, the head command will display the first 10 lines of the file so we'll create a file with 12 lines.

And if we want to only see 5 lines, we can use the -5 option.

```
[linux12@webminal.org~]$ cat hello3.txt hello3.txt hello3.txt hello3.txt > hello12.txt
[linux12@webminal.org~]$ head hello12.txt
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
[linux12@webminal.org~]$
```

Many times, we want to see a smaller number of lines and we can use the -n option along with the head command where n is the number of lines to show. Let's try displaying on the first 5 lines.

```
[linux12@webminal.org~]$ head -5 hello12.txt
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
Hi Linux CLI experimenters
[linux12@webminal.org~]$
```

Exercise:

Create a text document using either the touch or cat command. You can go to your desktop environment and edit it by hand if you don't want to try doing it in the terminal. Then, create a document with 25 lines. You don't have to write anything on the lines if you don't want to; just make sure that you number them 1-25.

Use the **head** command to view the first ten lines of your document in the terminal.

Next, add an option, such as -n, and the number of lines you want to view. For instance, try using -18 to view the first 18 lines of the document in the terminal.

Experiment with options you've learned so far in this book and see if any of them work with the head command. If so, what are they and what do they do? Write your findings in your notebook.

#

tail

Like the **head** command, the **tail** command displays a certain number of lines from a file as standard output but is the opposite of the **head** command. The **tail** command returns the last ten lines of a document or other file by default. Its usage is similar to the **head** command and you can use the -n option.

To use this command, type tail and the name or the file or directory that you want it to read and display.

For example, imagine that you only need the code in the last five lines of a long and complicated file. Type tail and the name of the file. Like the head command, you can also alter the number of lines it prints out using the -n option.

You can also use the -c option to limit the number of bytes, as opposed to lines. For example, if you only wanted it to display 12 bytes of information, you might type -c 12.

Explanation:

Like the **head** command, the **tail** command has a range of uses, though it mostly serves as a means of helping you quickly find important information. It can become tedious to print a long file to the terminal and scan through it manually for things like coding errors, update problems, or other inconsistencies.

Many times, the error in a log file will be at the end so the **tail** command is perfect for seeing what caused an error.

Exercise:

Go back to the document you created in the previous exercise and use the **tail** command and the -n option to print out only the last five lines of your document.

Experiment with the -c option. Type -c and a number and see what happens. Record your findings in your notebook.

The Powerful Built-In vi Editor

Most Linux distributions include some built-in editors such as vi, nano or emacs. The vi editor is very light-weight and is commonly built-in even on space-constrained Linux distributions such as the Raspberry Pi.

vi & vim

Vi is a text editor for Unix and Unix-like systems, such as Linux. Enter vi by typing vi into the command line and pressing **Enter** . You may be taken to vim, which is okay because vim is a lot like vi with a few extra features.

vim

Typing vim will also allow you to use a text editor in the terminal. The abbreviation vim stands for "vi improved" in Linux, and it offers a few features not found in previous versions of the vi text editor.

vimtutor

This section will provide an overview of the vi and vim editors but a vim tutorial is built into most Linux distributions and can be viewed by typing vimtutor. The tutorial will take less than an hour and has been helping people learn to use vi and vim for many years.

vi/vim Basics

The vi/vim editor has 3 modes, command mode, insert mode, and visual mode.

- Command Mode - This is the default mode and lets you move around in the editor, exit the editor, and search for text. Press the **Escape** key to return to Command Mode from the other modes.
- Insert Mode - This mode lets you insert text into the document. To get into this mode, you first need to be in Command Mode and then press the i key to enter Insert Mode. Press the **Escape** key at any time to exit Insert Mode.
- Visual Mode - This mode is used for selecting text in the document. To get into this mode, you first need to be in Command Mode and then press the v key to enter Visual Mode. Press the **Escape** key at any time to exit Visual Mode.

Quitting vi or vim

To quit, type colon and then the letter q ":q" into the terminal and press **Enter** . Like every other program run in the terminal, vi and vim require that you type a different command into the terminal to quit. Typing **Escape** or **Delete** will not yield any results.

Exercise:

Open vim by typing vim and the name of the file you want to edit. For this exercise, we will edit that file we were working on in the previous example. You should be able to use the arrow keys to move the cursor to where you want to begin editing. Press i and then the **Enter** or **Return** key to enable editing. Add a sentence such as "I'm using vim" to this document. Next, save the document. Press **Escape** to go back into "Command" mode, then type ":q" to quit. It may take a few tries before you feel comfortable using the editor and entering the key combinations.

More vim

Other things you can do with vim include viewing and editing two documents or files at once, copying, and pasting.

Vertical Split

To view two documents or files at once, type in vsplit and the name of the file. In some cases, you may have to type in the absolute path. Once you do this, you can now view and edit both documents. This may come in handy if you are editing a file's code and you need certain lines of the two to match.

Copying

If you want to copy text in vim, enter Visual mode and select the text and type y (short for yank). Make sure you are in Visual mode, or this may not work.

Cutting

If you want to cut text in vim, enter Visual mode and select the text and type "d." Make sure you are in Visual mode or this may not work.

Pasting

To paste, type p. Make sure you are in Visual mode or this may not work. The program should let you know what mode it's in by displaying it at the bottom.

Exercise:

Practice opening up two files at once and edit them to say the same thing. Then

try cutting, copying, and pasting information into your document.

Play around with what you've learned and try to master the vim text editor. Remember, it might take a little practice to get the commands right when you first start using this editor.

Chapter Summary

- In this chapter, we covered some basic commands, including commands that may help you change directories, list the files in a directory, view the path of the directory you are currently working in, clear the screen of text and already executed commands, view a file's type, move files, remove files, remove empty and full directories, create blank files, concatenate files, print file contents to the screen of your terminal, and view a set number of lines or bytes in a file.

- We also practiced some helpful exercises that may aid you in remembering those commands.

CHAPTER SIX: THE LINUX COMMAND LINE - MORE ADVANCED COMMANDS

The Command Line - More Advanced Commands

Command	Description
locate	Locate file
find	Find file or files
grep	Global Regular Expression Parser
tar	Tape Archive
gzip	Zip compresssion
zcat	Read contents of Zip file
wc	Word Count
history	Display command History
tree	Show Tree representation
alias	Create a shortcode for a command
curl	Download a file
cmp	Compare
diff	Difference between files
which	Which command executes
ping	See if anyone out there
man	Manual
help	Help on command

Searching

Wildcards

In Linux, "wildcards" are symbols that can substitute for other information when performing a search. Three wildcard symbols are used quite frequently in Linux, each with their own unique functions: the star wildcard * or asterisk symbol; the question mark wildcard or ?; and the square bracket wildcard or [ ].

*	Match one or more preceding terms
.	Match one instance of any character except newline
?	Match zero or one of the preceding character
[ ]	Match a character in the set

The Star Wildcard -> *

The star wildcard, or asterisk symbol, is used to match one or more instances of any character when searching for information. For example, if you type ‘find . -name "my_test*"’ in the terminal and press **Enter** , your search results will list all files that start with my_test in the directory in which you are currently working.

The Question Mark Wildcard -> ?

The question mark wildcard, or ? Is used to match one character in the search. For example, if you wanted to search for every instance of a file that was only five characters long, you would type, find . -name "?????" and press **Enter** . Note the difference between ? and * - the * matches 1 or more characters while the ? only matches 1 character.

The Square Brackets Wildcard -> [ ]

You can use the square brackets wildcard, or [ ], to search for specific information contained in filenames or directories. For instance, if you wanted to find every file in a specific directory that started with the letter "a," "b," or "c" in lowercase, you could enter, ‘find . -name "[a,b,c]*"’ into the command prompt. The brackets match a, b or c while the * covers any other characters in the name. If you only wanted the file named a, b, or c, you would use ‘find . -name "[a,b,c]"’ and only those characters would be matched.

Many other things can also be done with the square brackets wildcard command, such as finding files within a certain range, or even sets of ranges. For instance,

if you wanted to search for every file that began with a letter, you could type, ‘find . -name "[a-z]*"’ and press **Enter** . The search results should yield a list of every file that has a name that starts with a lowercase letter.

Exercise:

Use the star wildcard * to find every instance of a file containing a certain word in your Desktop folder.

Try using the question mark wildcard ? to find every instance of a file with a certain number of letters.

Use the square brackets wildcard [ ] to find every instance of a file that starts with the letter "a."

Pair these three wildcards with three commands you learned in this chapter. Record your findings in your notebook.

#

locate

To use the **locate** command to find a file, type locate and the name of the file you want to find; then, press **Enter** . The search results should list the directory and path in which the file is found.

Example:

Let's pretend you are searching for a file that you named "codewords2019.txt", but you can't find all of them. Using the **locate** command should find that file for you while also giving you their directory path, so you can access and modify it without looking through a bunch of unrelated files.

Explanation:

The **locate** command helps you find instances of a file name and may be useful if you have multiple files of the same name and do not know how to access them. Using it will result in a list of files that contain your keywords and the paths associated with the listed files.

Exercise:

Create a file or search for one already created using the locate command. Record your findings in your notebook.

Regular Expressions:

If you need to narrow down your search, you may be able to do so by using regular expressions such as grep. We will be covering these wildcards later in this chapter.

#

find

The **find** command can be used like the **ls** command to search for all files in the current working directory. Just type **find** and no other arguments. You can also type in the direct path if you want to have it list other files in another directory.

It can also be used for other things, such as what the next few sections will cover.

Searching for Files

Usage: find <path to start searching from> -name <name to search for>

You can search for files with a specific name as follows:

```
[linux12@webminal.org~]$ find / -name "my_filename.txt"

my_filename.txt

[linux12@webminal.org~]$
```

Will find all the files named "my_filename.txt" in the root directory. This command becomes very powerful when paired with wildcards.

This command can also be used to search for files of a specific type. Type find -name "*.[the type of file you want to be found]".

For example, if you want to see all of the ".jpg" files in the current directory and subdirectories, you would type, find . -name "*.jpg" and **Enter**

```
[linux12@webminal.org~]$ find . -name "*.jpg"

photo_jpg.jpg

[linux12@webminal.org~]$
```

You can also use this command when searching for files that are not of a certain type. For example, if you entered find . -not -name "*.jpg", you would get a list

of all files that are not .jpg files.

Exercise:

Use the find command to locate a list of files of a specific type. Write down the file you want to use in the next part of this exercise, then copy the absolute path of the file in case you need to enter it.

Next, search for that specific file. Was Linux able to find it? Record your findings in your notebook.

#

grep

The **grep** command is the global regular expression parser and it looks for a pattern in a file. To use it, type grep followed by the command or string you want to find, as well as the location you want it to search for the command or string.

For instance, if you wanted to find all files in a particular directory starting with a specific word or letter, you could use the **grep** command by typing “grep [name of the file you want to find] [name of the directory where you want to find it]”.

If you wanted to find user information for a user named James, you could type something like the following and get the resulting print out.

```
[linux12@webminal.org~]$ grep james /etc/passwd

james:x:1000:1000:james,,,:/home/james:/bin/bash

[linux12@webminal.org~]$
```

Explanation:

grep stands for "global regular expression parser," and the name comes from when software was simple and one of its main concerns was saving space.

Exercise:

Use **grep** to search for a word. In this scenario, we will be searching for every file with the word "photos." Type **grep** -w "photos" file, then search for a file you have that starts with a certain word using the "**grep** " command. If you don't

have a file, either rename an existing file or create one with the touch command.

Next, search for one of the users on your computer or network; if you are using a virtual machine, that might be only you. In that case, search for your username. Record your findings in your notebook.

Archive And Compression

tar

In Linux, **tar** stands for "tape archive." You can use this command to create archive files and compress and store data, so it takes up less space on your drive. Just type **tar** , choose the option you want to use, name the file, and decide where you want to store it.

In this example, let's say you wanted to create a **tar** file named "abc" and from the directory called "docs." To do this, you would type something like tar abc.tar usr/home/docs. If you did it correctly, you would have a .tar file called "abc.tar" that includes the contents of the docs directory in the current directory.

Explanation:

The tar command helps you save space, which can come in handy if you have a computer or network with limited storage space due to usage as a server or multi-user machine or network.

Exercise:

Create a new file using the touch command, then change it into a .tar file.

#

gzip

The **gzip** command zips a file, and you can use the **gzip** command in a similar way to the **tar** command. Like tar, it is used to compress or expand files. To use this feature, type **gzip** and the filename into the terminal, then press **Enter** .

By default, the **gzip** command deletes the original file once it creates the zipped version. You can prevent this, however, by typing -k before the filename. For example, if you did not want Linux to delete the original file, you could type “gzip -k filename”.

Exercise:

Create a zipped text file using gzip.

#

zcat

If you want to read files that were gzipped without unzipping them and potentially wasting time and space in the process, you can use the **zcat** command. This command will allow you to read gzipped files from the terminal. To use it, type **zcat** and the name of the gzipped file you want to have a look at. Then, press **Enter** .

If you've done this correctly, you should be able to view the contents of the file from the terminal without having to uncompress the file.

Exercise:

Read a zipped text file without unzipping it first using the zcat command.

Useful Commands

wc

The **wc** , or "word count," command counts the number of words, letters, and lines of the information it is told to count. To use it, you would type **wc** and the name of the file or directory in which you want the word count performed.

For example, let's say we wanted to get the word count of file1.txt. We could type “wc file1.txt”. The results should give us three numbers--the first is the number of lines, the second is the number of words, and the third is the number of bytes. If you only wanted to know one of these things, you could use other options.

We can also get the number of lines by typing “wc -l file.1.txt”.

```
[linux12@webminal.org~]$ wc -l file.txt
1  4  27  hello.txt
[linux12@webminal.org~]$ wc -l file.txt
1  hello.txt
[linux12@webminal.org~]$
```

Replacing the -l in the previous example with a -w will give you just the word

count.

Replacing the -w with -c would give us the number of bytes.

You can also use the --help command for a list of even more options.

Exercise:

Find the word count, number of lines, and number of bytes of a file of your choice. Try to use a different file than the ones you've used for previous exercises.

Compare the word count of two different files, then write your findings in your notebook.

#

history

Linux allows you to view the commands that have been typed in the terminal previously. Type **history** and press **Enter** to see the command **history** . By default, Linux will list the last 500 commands you typed into the terminal. You can even combine this command with other commands, such as the **tail** command, to make it more useful.

Sometimes you may need a reminder of how exactly to write a command. Instead of looking them up all the time or memorizing them, you can use Linux's built-in features to make that process much easier. Some commands, such as the **help** command, the man command, and the **history** command, all fall into this category.

Entering the history command allows you to view the past 500 commands you entered into the terminal, which will come in handy if you don't remember exactly what you typed and don't want to or can't afford to waste time looking it up on the Internet.

You can also find recent commands by pressing the **Up arrow key** .

```
[linux12@webminal.org~]$ history
1  ls -al
2  cat hello.txt
3  cat hello.txt  hello.txt  hello.txt
4  cat hello.txt  hello.txt  hello.txt  >hello3.txt
5  history
[linux12@webminal.org~]$
```

Exercise:

Use the history command to display your last 500 commands.

Next, jump to the Magic of Piping section and the use piping with another command you learned earlier in this chapter to make the history command display only the last 25 commands you used. Write down which commands you entered in the Linux terminal to make it work.

#

tree

Do you remember using the **tree** command earlier in this book? You can use the **tree** command if you want to know what subdirectories and files are contained in the folder in which you are working. Type **tree** into the command line, then press **Enter** .

This may remind you of the ls command, but it is much more detailed, as it shows not only each folder a directory contains but also each program file contained inside those folders as well.

Exercise:

Use the tree command to generate a detailed list of files and folders in your current working directory. Then, do the same thing using the ls command. Write how they are different in your notebook. What might you use the ls command for that you would not use the tree command for? What might you use the tree command for that you couldn't use the list command for?

#

alias

This is a shortcut for commands that you use frequently. Some people use an **alias** to change the default behavior of a command. For example, we could assign an **alias** to the remove rm command so that it asks us to confirm the deletion of a file (the rm -l where that is a lowercase L not a #1). To use the **alias** command, just type the new command you created.

Example:

A common **alias** is ll which will call the ls -al to show all files in long-form.

```
[linux12@webminal.org~]$ alias ll="ls -al"
[linux12@webminal.org~]$ ll
total 5972
drwx------.   4 linux12 linux12   4096 Feb  4 03:13 .
drwxr-xr-x. 172128 root   root   4087808 Feb  4 03:21 ..
-rw-r--r--.   1 linux12 linux12   2785 Feb  3 04:04 .bash_history
-rw-r--r--.   1 linux12 linux12     18 Dec  7  2016 .bash_logout
-rw-r--r--.   1 linux12 linux12    193 Dec  7  2016 .bash_profile
-rw-r--r--.   1 linux12 linux12    231 Dec  7  2016 .bashrc
drwxrwxr-x.   3 linux12 linux12     22 Feb  2 21:50 Desktop
-rw-r--r--.   1 linux12 linux12    334 Sep 20  2017 .emacs
-rw-rw-r--.   1 linux12 linux12      0 Feb  2 21:33 photos.png
-rw-rw-r--.   1 linux12 linux12     31 Feb  2 20:12 textfile.txt
-rw-r--r--.   1 linux12 linux12    658 Aug  2  2017 .zshrc
[linux12@webminal.org~]$
```

Don't worry about messing anything up; your custom commands will not be available in your next session unless you edit your shell configuration profile.

You can look up how to edit your shell configuration profile, but make sure you are well versed in Linux programming and commands before you do that, as it is not a task that is recommended for beginners.

It is possible to mess a few things up when creating custom commands, so be

careful and don't opt to make them permanent until you have tested it and know it works.

Exercise:

Create three custom commands and then run them. Record your findings in your notebook. What were your commands, and what did they help you do? What potential uses do you see for setting aliases in the future?

#

curl

The **curl** command can be used to download files from the internet using the terminal in the absence of a GUI. To use this command, type curl, the option that you want the machine to perform, and the URL of the information you want to retrieve. You can also use the curl command to upload files to the Internet.

Example:

If you wanted to download a picture of the Biggest Ball of Twine using the **curl** command, you would enter

```
[linux12@webminal.org~]$curl
https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/TwineBallCawke
rKs.jpg/450px-TwineBallCawkerKs.jpg

[linux12@webminal.org~]$
```

Doing so would download a file of the same name as the file you typed in.

If you want to use **curl** to view the content of a website, type **curl** [site whose file content you want to view]. The result should be the HTTP source code.

To upload a file, you can type "curl [file] [upload address]." For example, if I wanted to upload a file called "photos" to a local server called "server," I might type,

```
[linux12@webminal.org~]$curl -F 'photos@home/Desktop/Pictures/photos.jpg'
http: // server/upload

[linux12@webminal.org~]$
```

Exercise:

Download a file and save it on your desktop using the curl command.

View the contents of a website of your choice using the curl command.

Write your findings in your notebook.

#

diff

The diff command compares two similar files and prints out the parts that are different, making changing the files to where they are identical much easier. This command is easy to use -- just open your Linux terminal and type diff, then the name of the two files you want to be compared, separated by a space. For example, you can type diff file1 file2

#

cmp

If you want to compare two files byte by byte, you can use the cmp command. Unlike the diff command, this command will print out which bytes and line numbers are different in the files being compared.

Exercise:

Practice using the diff and cmp commands.

For the **diff** exercise:

Open a word processing program on the GUI and make a short list of items. Then, save it as a .txt file and copy the list. Paste it in another document on the word processing program and change one word. Name these files something recognizable, and then go back to the terminal.

Example:

Document 1	Document 2
1. Rosemary	1. Rosemary

1. Curry	2. Thyme

Type **diff** into the terminal and enter the names of the two files you created. The results should print to the screen at the terminal. Record your findings in your notebook.

For the **cmp** exercise:

Open another word processing document on the GUI and make two documents, each with a shortlist of the same items. This time, don't change any of the text in one of the documents except for one number on the list. You should have two nearly identical documents, except one with a different number on the list.

Example:

Document 1	Document 2
1. Rosemary	1. Rosemary
2. Curry	1. Curry

Run the **cmp** command and record your findings.

#

which

The Linux **which** command can aid you in locating where your important programs are installed by providing the absolute path of the file. To use this command, type **which** , followed by the name of the program you want to find. For example, if you are looking for the path and directory where the executable file for the internet browser "Firefox" is stored, you would type “which firefox” and press the **Enter** key in the terminal.

```
[linux12@webminal.org~]$which firefox
/usr/bin/firefox
[linux12@webminal.org~]$
```

The directory and its path should print to the screen, which should look something like /usr/bin/firefox. We can also use this to find multiple files by typing which followed by the name of each file you want to locate, separated by a space. For example, entering which firefox gimp would give you the paths and directories of both the Firefox internet browser and the Gimp photo editing program.

Exercise:

Use the **which** command to locate a file you created in a previous exercise, then write down the results in your notebook.

#

ping

The **ping** command lets you check whether or not a remote host is responding. Just type in **ping** , followed by the URL of the website you are trying to reach. For example, if you wanted to ping Google, you would type, **ping** www.google.com.

Exercises:

ping a website of your choice. Use the **Ctrl+c** keyboard shortcut to escape the loop, then record your findings.

Help Commands

man

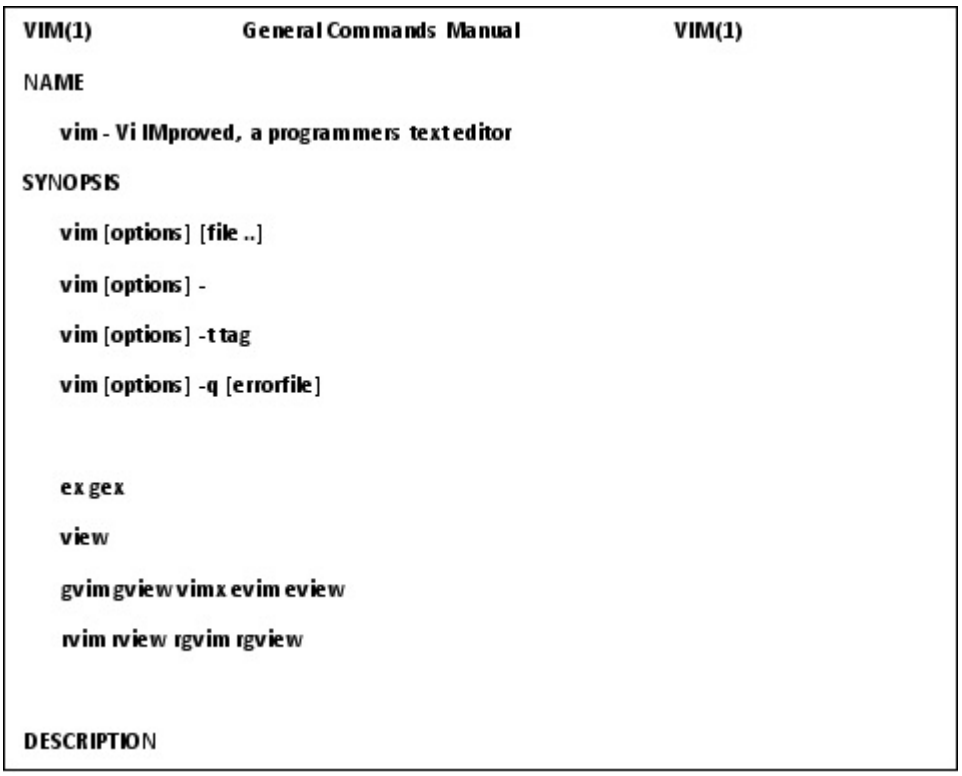
If you type in **man** in the terminal followed by the command you want to be described, you'll get a list of uses for the command. For example, if you typed **man** touch, you would get a list of uses for the **touch** command. This comes in handy if you need to know what a command is in a pinch and don't have time to look it up.

Typing **man** for the manual page command will give you a lot more information

about the command you may want to use, including syntax for executing successful versions of the command you want to use. You will also get information on what the command does.

Exercise:

Look up the **man vim** command by typing “man vim” at a command prompt. You should get something like below:



```
VIM(1)                General Commands Manual                VIM(1)

NAME

    vim - Vi IMproved, a programmers text editor

SYNOPSIS

    vim [options] [file ..]

    vim [options] -

    vim [options] -t tag

    vim [options] -q [errorfile]


    ex gex

    view

    gvim gview vimx evim eview

    rvim rview rgvim rgview


DESCRIPTION
```

And the Description follows the above:

```
DESCRIPTION
    Vim is a text editor that is upwards compatible to Vi. It can be used to edit all
    kinds of plain text. It is especially useful for editing programs.
    There are a lot of enhancements above Vi: multi level undo, multi windows and buf-
    fers, syntax highlighting, command line editing, filename completion, on-line help,
    visual selection, etc.. See ":help vi_diff.txt" for a summary of the differences
    between Vim and Vi.
    While running Vim a lot of help can be obtained from the on-line help system, with
    the ":help" command. See the ON-LINE HELP section below.
    Most often Vim is started to edit a single file with the command
      vim file
    More generally Vim is started with:
      vim [options] [filelist]
```

Further Explanation:

man pages = manual pages

Back when programming was just beginning, remembering a certain command syntax wasn't as easy as hopping on the Internet and looking it up. Linux was created from Unix and computers were around before the Internet was widely available. It made more sense to build a list of command explanations into the system itself.

Since this system was invented before internet access and reliable search engines were available, the man command provided the user with a convenient way to look up information about different commands.

Today, man pages are still useful, especially if you are working in a terminal that has no GUI and is not connected to the internet. No matter where you are, you can still get help finding the proper commands and what they do.

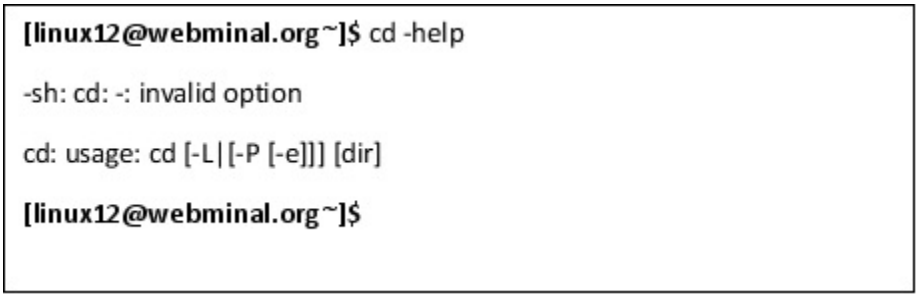
If you are unsure of how to use a certain command, type man, then the command that you are having trouble with a space in the middle. An explanation of the command and how it is to be used will show up on the screen.

Type clear if you want the terminal to stop displaying the man page text.

#

help

The **help** command functions like the **man** command and it can sometimes be used to find options to use with commands. The **help** command can also remind you how a command is supposed to be used. To use it, type the command and then -help. For instance to see the help for cd, type cd -help



The Magic of Piping

Let's talk a little bit about piping, revisit grep, and go over the various options that can be used with this helpful command.

If you've ever browsed a Linux help forum, you might have come across people talking about "piping" certain programs. It's not a complicated process that only computer programmers can master; in fact, piping isn't much more complicated than any other Linux command.

One important aspect of Linux command lines is the ability to "pipe" commands. I personally consider this Linux's greatest strength.

The Unix philosophy is to have each program do one thing well. **Piping** creates a small chain of commands. **The output from one command is used as input for the next .** For example, if you searched for all of the .doc files in a certain directory, then directed Linux to grep those files to find the desired string, that would be an example of piping.

Piping can be described as the process of redirecting standard output to another command, program, or process to be worked on further. Let's say you want documents sorted, but you only want to sort the ones that have "cheese" in the title. You may consider using a pipe command to help you give those instructions to the terminal.

Let's make a simple piping command.

Find the "| " key on your keyboard--it is usually **Shift+** . This is the symbol you will use to create a piping command. Open up your terminal and type the following: ls | grep txt. Write your findings in your notebook.

```
[linux12@webminal.org~]$ ls -l
total 24
drwxrwxr-x. 3 linux12 linux12   22 Feb  2 21:50 Desktop
-rw-rw-r--. 1 linux12 linux12   27 Feb  2 22:08 hello.txt
-rw-rw-r--. 1 linux12 linux12 6084 Feb  2 22:30 longtext.txt
drwxrwxr-x. 2 linux12 linux12    6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12    0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12   31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$ ls | grep txt
hello.txt
longtext.txt
textfile.txt
[linux12@webminal.org~]$
```

Exercise:

Create your own pipe and enter the command in the terminal. Write down your findings in your notebook.

Chapter Summary

- In this chapter, we covered more advanced Linux commands, including searching using the locate, which, and find commands. We also learned about the grep command; how to find the word count, line count, number of bytes; and how to use wildcards.
- We also covered other useful commands, such as zipping files, history, tree, alias, curl, diff, cmp, and pinging.
- We also covered some helpful commands, including man and the help commands.

CHAPTER SEVEN: LINUX SYSTEM AND USER MANAGEMENT

Important System Management Commands

Command	Description
df	Display filesystem space
du	Disk usage
free	Free memory usage display
w	Which users are logged in?
whoami	Whoami logged in as?
hostname	List the hostname of the machine
uname	Print kernel information
quota	Quotas and limits of users
passwd	Set/change a users password
chage	Force user to change password
adduser	Add new user
usermod	Modify a user
addgroup	Add a new group
userdel	Delete user
delgroup	Delete a group

System Management Command

The following commands are commonly used by IT professionals for managing computer systems inside an organization.

#

df

The **df** command stands for "display file" or "file system space" and allows you to display file system space that is being used and is available. To use it, all you have to do is type **df** and the folder name. If you only type df, an overview as shown below is given.

```
[linux12@webminal.org~]$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda1      103117024 47921020  50983664  49% /
devtmpfs         7425888       0   7425888   0% /dev
tmpfs            7435592 1281488   6154104  18% /dev/shm
tmpfs            7435592  733024   6702568  10% /run
tmpfs            7435592       0   7435592   0% /sys/fs/cgroup
/dev/sdc1       31440900 14626824  16814076  47% /home
/dev/sdb        10475520 3827180   6648340  37% /common_pool
[linux12@webminal.org~]$
```

#

du

The **du** command is the "disk usage command," which allows you to see disk utilization for files and commands. To use this command, type **du** followed by the directory. This command may come in handy if you need to recycle data space used by a program, although it may give you a rounded up estimate instead of actual, real-time usage.

A common way to type this is “du -ch” which gives a summary (the -c) and human-readable output (the -h) which append the K for Kilobytes, M for

Megabytes, or G for Gigabytes.

```
[linux12@webminal.org~]$ du
0     ./my_test
0     ./Desktop/my_photos
0     ./Desktop
60    .
[linux12@webminal.org~]$ du -ch
0     ./my_test
0     ./Desktop/my_photos
0     ./Desktop
60K   .
60K   total
[linux12@webminal.org~]$
```

#

free

Use the **free** command to find out how much memory and system RAM is available. This may come in handy if you need to know if a program is eating up system resources or how much memory a program is using.

```
[linux12@webminal.org~]$ free
              total        used        free      shared  buff/cache   available
Mem:       14871184     2396968     3561344     2014516     8912872     9994256
Swap:             0           0           0
[linux12@webminal.org~]$
```

#

w

The **w** command shows information about users who are currently logged on to the system, including what they are doing. This potentially comes in handy if you want to check employee productivity or if you want to check which employee is running a task that puts a heavy load on the system memory.

If you work in IT, this command may be especially helpful for you because it may prevent you from working on the machine or installing programs during times where employee workloads are already putting a strain on the memory of the machine.

#

whoami

The **whoami** command quickly shows you which username and account you are logged in as. This may prove helpful if you are not sure what permissions your user account has and what commands you have the authority to run. It could also help you if you are logging yourself into different accounts with different permissions while testing features on the system.

To use this command, type **whoami** into the terminal and press **Enter** . You should see a readout with your username.

```
[linux12@webminal.org~]$ whoami
linux12
[linux12@webminal.org~]$
```

#

hostname

You can use the **hostname** command to find out the name of the host. You can also use it to print out the IP address with the -i option. In the result below, the ip address of 127.0.0.1 indicates a localhost meaning "this computer".

```
[linux12@webminal.org~]$ hostname
server-1.localdomain
[linux12@webminal.org~]$ hostname -i
::1%1 127.0.0.1
[linux12@webminal.org~]$
```

#

uname

This will give you a printout of information about the operating system and the machine it is installed on. Use the -a option to see all the kernel information including build date kernel version info.

```
[linux12@webminal.org~]$ uname
Linux
[linux12@webminal.org~]$ uname -a
Linux server-1.localdomain 3.10.0-514.16.1.el7.x86_64 #1 SMP Wed Apr 12
15:04:24 UTC 2017 x86_64 x86_64 x
86_64 GNU/Linux
[linux12@webminal.org~]$
```

#

Typical User Management Commands

The root user has administrative privileges, which means that the user with access to the root account can do nearly anything in the system, even bypassing security checks for normal users without the root privilege. While the root user always has administrative privileges, other users can have admin privileges or just standard privileges.

Having administrative privileges is a good thing if used properly, but it can be harmful if used carelessly. Administrative/root users can mess up the operating

system or even harm others' accounts if they are not aware of what they are doing. That's why having root privileges is such as big deal in Linux.

Administrators, IT professionals, and anyone else who happens to have access to the root account have to make sure they enter the correct commands and know what they are doing to keep from accidentally deleting sensitive data or rendering programs non-functional. In addition to this, those with administrative privileges usually have the responsibility of looking after user accounts, fixing bugs, handling password issues, and maintaining a decent level of security.

Luckily, Linux has commands that make keeping track of data and users easier.

#

quota

Without a set data quota, users on a network could hog all the resources for themselves. They could start downloading unnecessary files and keeping bit and pieces of information, such as reports, instructional videos, and projects on their accounts as if it were their home machine, while not considering the needs of other users or the operating system.

With the **quota** command, limits for the user can easily be monitored. This command displays the user's disk usage and limits, and a user with administrator privileges can use this command to view the quotas of other users. To use this command, type, quota -s and the user ID of the user whose quota you want to check.

Editing Quotas

If you want to change the quota of a user or group, you can use the edquota command by typing edquota -u and a username or ID to edit the quota of an individual user. To edit the quota of a whole group, type edquota -g and the group name or group ID.

#

passwd

As mentioned previously, users with administrative privileges can do a lot with Linux if they know how. The following are a few more things you might have to do if your job grants you root user privileges.

Let's imagine that someone at work forgets his or her password. If you are the IT

person in charge, you should know how to properly deal with that situation. Unless the company you are working for has an enhanced security add-on system that doesn't grant true admin privileges, this problem can be easily remedied by typing

```
$ passwd <username>.
```

There may also be situations in which the user will need to change his or her password but has not changed it yet. In this case, you may want to execute a command that will force a user to change his or her password (see the chage command below).

The **passwd** command is used to change a user's password. This command can only be used by someone with administrator access. An administrator can also use it with other options to do a variety of other things, such as -w to warn a user before their password expires, or the -n option to set the minimum number of days a user must wait before changing his or her password.

Exercise:

Experiment with the passwd command and its options. Log in as the target user to test and write your findings in your notebook.

#

chage

The **chage** command can be used in Linux to force a user to change a password by typing “chage -d 0 <username>”. Like the setting in the previous example, this command will also force the user to change his or her password upon next login.

If you want to make it so the user who refuses to change his or her password will be locked out of his or her account, use the -E option with the chage command instead. It should look something like chage -E <date or days until expiration> <username>. This will lock the user out of his or her account until he or she has it reset by an administrator or root user.

These features are important because they can serve to automatically enforce policies your company may have about users changing their passwords, so important information is not compromised.

Exercise:

Use the **chage** command on a user you created, so the user will have to change his or her password. Log in as the user to test this.

#

adduser

The **adduser** command can be used to create and add new users. Type, **adduser** and the username you want the new user to have to create a new user.

Adding a Normal User

To add a normal user, type adduser and the username you want the new user to have. The user will automatically be assigned an ID number and granted all the privileges a normal user on the system has.

```
testuser@testuser-VirtualBox:~$ sudo adduser norman
[sudo] password for testuser:
Adding user `norman' ...
Adding new group `norman' (1001) ...
Adding new user `norman' (1001) with group `norman' ...
Creating home directory `/home/norman' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for norman
Enter the new value, or press ENTER for the default
        Full Name []: Norman Lastname
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
testuser@testuser-VirtualBox:~$
```

Adding an Administrative User

To add an administrative user with higher privileges like root access, follow the same process of adding a new normal user above. The difference is that the admin user is added to a group with higher privileges. For instance on Ubuntu, this is the sudo group.

```
testuser@testuser-VirtualBox:~$ groups norman
norman : norman
testuser@testuser-VirtualBox:~$ sudo adduser norman sudo
Adding user `norman' to group `sudo' ...
Adding user norman to group sudo
Done.
testuser@testuser-VirtualBox:~$ groups norman
norman : norman sudo
testuser@testuser-VirtualBox:~$
```

Adding a System User

System users are accounts created when certain programs are installed and they perform certain jobs and run certain processes. They typically don't belong to actual human users, but instead to programs that run in the background. They have limited access to the system, enough so they can perform their tasks with minimal human interference.

Assigning these programs group IDs and privileges controlled by the administrator is one way Linux keeps the system safe from malicious software and unruly programs. Contrast this to a Windows system, in which an installed program is typically given admin privileges and has a greater chance of harming the system.

To add a system user, type **adduser --system** . The user will automatically be assigned a user ID, or you can specify a user ID. Choosing a user ID of an existing user will cause the adduser program to exit and show a warning.

```
$ adduser --system
```

System users are placed in the nogroup group by default, but you can edit a system user's group ID with the gid option.

Force New Password

To force a user to change his or her password via a command at the terminal, type passwd --expire <username>, which will cause the user's password to

expire. The user will be prompted to change it upon the next login.

```
$ passwd--expire<username>
```

Exercise:

Create a new user using the **adduser** command. You can delete this user account using the **deluser** command later, or you can keep the account for future exercises in this chapter and extra practice.

#

usermod

Another powerful command you might find yourself needing to use as an admin is the usermod command. Use this command to change various details about a user's account, including their group IDs, whether or not the user's account is active, and a few other things.

Lock/Unlock

One thing you can do with the **usermod** command is "lock" and "unlock" user accounts. Let's pretend that you need to lock a user's account because he or she no longer uses that account. Simply type usermod -L to lock the account and usermod -U whenever you are ready to unlock the account.

Adding an Administrative User

To give an normal user higher privileges like root access, use the usermod command to add them to a higher privilege group. For instance on Ubuntu, this is the sudo group. You learned how to add the user to the sudo group using the **adduser** command, but the **usermod** command can also be used.

```
testuser@testuser-VirtualBox:~$ sudo usermod -aG sudo norman
testuser@testuser-VirtualBox:~$ groups norman
norman : norman sudo
testuser@testuser-VirtualBox:~$
```

Expiry Date

You can also use the option -e to set an expiry date on a user account. This may come in handy when reminding a user to change his or her password.

Exercise:

Create a test user account and lock the password. Then, try to log in as the test user.

Remember, use -L to lock the user password so that it cannot be used. Use -U to unlock the password.

Exercise:

Give the password of your imaginary user an expiration date, then write your findings in your notebook.

#

addgroup

The **addgroup** command is similar to the **adduser** command, except users who belong to a group can share documents and files and may have access to certain permissions and files that the rest of the users don't. It can also be a way to evenly set permissions for some users.

For example, let's pretend you have a group called "employees." Everyone added to the employee group can access project files and the chat feature to communicate with their teammates, but they do not have administrator rights and cannot go into the system and change crucial settings, re-assign programs to new directories, change others' passwords, allot themselves more data, among other things.

```
testuser@testuser-VirtualBox:~$ groups norman
norman : norman sudo
testuser@testuser-VirtualBox:~$ sudo addgroup employees
Adding group `employees' (GID 1002) ...
Done.
testuser@testuser-VirtualBox:~$ sudo usermod -aG employees norman
testuser@testuser-VirtualBox:~$ groups norman
norman : norman sudo employees
testuser@testuser-VirtualBox:~$
```

Exercise:

Create a group and add the user you created in the previous exercise to it.

#

userdel

If you accidentally create an extra user, need to get rid of a test user account, or need to get rid of a user account for other reasons, you can do so with the **userdel** command. If you want to force a delete of a user account, type userdel -f <username>. This will delete the account, even if the user is currently logged in (although that would be mean). If you want to make sure the account is gone, type deluser <username> --remove-all-files.

#

delgroup

Deleting a group is similar to deleting a user. If you want to delete a group, type delgroup and the group you want to be deleted. Keep in mind that you cannot remove a user from his or her primary group; if you want to delete a user from his or her primary group, you can delete the user first.

Exercise:

Create a new user using the adduser command. Then, create a new group using the addgroup command. Place the new user in the group, then remove the group and the user.

You can also use deluser with the --group option to delete a group.

Exercise:

Practice all the commands covered in this chapter and pair them with some of the mentioned options. Record all your findings in your notebook.

Generate a list of questions for further study based on your list, then record where or how you think you can gather this information.

Chapter Summary

In this chapter, we went over:

- Managing users and passwords.
- Setting data quotas for users and administrators.

CHAPTER EIGHT: UNDERSTANDING LINUX SECURITY

Linux is different, not just in how it is built, but also the way that it's maintained. Unlike Windows, which may need an update every week to remain stable, Linux operating systems are built to be stable and run with minimal patching and interference.

There's also a dedicated community of volunteers, many of which are computer programmers, who are constantly checking the software to make sure it's running properly. Linux systems also have built-in features that make it secure, which makes it incredibly difficult to hack into. With Linux systems, you would have very little use for virus protection.

However, have you ever wondered why Linux is so secure?

Like some other more secure operating systems, like Chrome OS, Linux has a closed file system and doesn't allow content from unauthorized sources or users. The concept is about the same as apps in the Apple or Chrome store--if you've ever used one of these systems, you've probably noticed that you can't just download any file from the internet and automatically use it one your computer, even if it's the proper format.

This is because Apple and Chrome OS not only check for the proper format, but they also have other security measures in place to make sure you don't accidentally download files that could potentially harm your computer. This is intended to keep your computer safe from viruses, malware, and spyware; the downside is that it can also make your software choices a bit more limited.

You may have to find alternative apps that will run on your OS instead of downloading familiar software to do the same job; for example, on a Chrome OS, you'll have much less trouble running Google Docs than Microsoft Office.

Fortunately, the Linux repository has lots of alternative software programs to choose from, and if you know what you want to do, you shouldn't have a lot of trouble choosing an alternative. Most popular software has other versions that are available to Linux users.

The Underpinning of Linux Security

Administrator/Root User

Linux users typically fall into two categories in regards to permissions - admin users and normal users. Admin users are commonly the most powerful users on the Linux system, having root access to every part of the operating system and being able to execute commands that could damage the operating system if not used properly.

Admin/root users can pretty much do anything they want on the computer, including viewing and changing other user accounts. The root user can access anyone's home directory and data, and they are usually an administrator, as they need to be able to create, delete, and manage all users on the system.

Normal Users

User accounts are the accounts that belong to people who are authorized to use the machine but may not have all the access an administrator has. A typical person can have access to a Linux machine as a user and not have full administrator privileges. The user can perform the tasks he or she is authorized to do while minimizing the chances that they will mess up the operating system.

Exercise:

Explain how a root user is different from other users. Write your answers in your notebook before going on to the next section.

sudo

In Ubuntu, users may need to use a special command called sudo to gain root user access to certain files or privileges, such as downloading files or making major changes to the operating system.

As explained earlier in this book, sudo is short for "super user do."

In Ubuntu Linux, users are not in root user mode. For example, you may not be able to read or change some of the files in the root directory. The extra step of typing sudo in front of the command is one extra safeguard against accidentally damaging or deleting something important.

If you run into anything like this, you can try to use the sudo command to get access and run programs as the root user or administrator. Without typing sudo, you may not have the necessary permissions to run programs that only have administrator privileges and you may not be able to download files. See the example below to see how adduser on Ubuntu needs a sudo prefix.

```
testuser@testuser-VirtualBox:~$ adduser norman sudo
adduser: Only root may add a user or group to the system.
testuser@testuser-VirtualBox:~$ sudo adduser norman sudo
Adding user `norman' to group `sudo' ...
Adding user norman to group sudo
Done.
testuser@testuser-VirtualBox:~$ groups norman
norman : norman sudo
testuser@testuser-VirtualBox:~$
```

apt-get

Explanation:

Repositories are places on the internet that house Linux programs. You don't need to use a USB stick or memory card or type in a special code to install the software. Instead, it's a lot like visiting the app store on a Chromebook or Kindle, except done at the command line.

All you need to do is type in

$ sudo apt-get install <software_name>

Where <software_name> is the name of the program you want to download.

To remove the software, just type in

$ sudo apt-get remove <software_name>

Where <software_name> is the name of the program you want to remove.

To update, type in sudo apt-get upgrade and the name of the program you want to update.

Here is a quick list of repo commands. Use your favorite search engine such as duckduckgo to discover all the variations available.

- apt-get update will update the list of repos.
- apt-get upgrade will update the program or all the installed programs if a single program is not specified.
- apt-cache search will search for a program in the repo.

- apt-get install will install the program.

Let’s begin by updating the packages.

```
testuser@testuser-VirtualBox:~$ sudo apt-get update
Hit:1 https://download.sublimetext.com apt/stable/ InRelease
Ign:2 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:3 http://dl.google.com/linux/chrome/deb stable Release
Hit:5 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Get:6 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Fetched 2,077 kB in 27s (75.6 kB/s)
Reading package lists... Done
testuser@testuser-VirtualBox:~$
```

Now let’s search for the Python3 package.

```
testuser@testuser-VirtualBox:~$ sudo apt-cache search python3-wget
python3-wget - pure Python download utility for Python 3
testuser@testuser-VirtualBox:~$ sudo apt-get install python3-wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  python3-wget
0 upgraded, 1 newly installed, 0 to remove and 10 not upgraded.
 (Reading database ... 235142 files and directories currently installed.)
Preparing to unpack .../python3-wget_3.2-1_all.deb ...
Unpacking python3-wget (3.2-1) ...
Setting up python3-wget (3.2-1) ...
testuser@testuser-VirtualBox:~$
```

Files and Directories

As mentioned earlier, not every user has administrative permissions. In some cases, a user's access to certain files may be limited. Certain commands in Linux can grant permissions to users that they wouldn't normally have, such as read, write, and execute. These commands can also change a user's permissions on certain files.

Let's take a look at the ls -al command again.

```
[linux12@webminal.org~]$ ls -al
total 5932
drwx------.   4 linux12 linux12    4096 Feb  2 19:37 .
drwxr-xr-x. 171965 root   root    4083712 Feb  2 19:35 ..
-rw-r--r--.   1 linux12 linux12       0 May 31  2017 .bash_history
-rw-r--r--.   1 linux12 linux12      18 Dec  7  2016 .bash_logout
-rw-r--r--.   1 linux12 linux12     193 Dec  7  2016 .bash_profile
-rw-r--r--.   1 linux12 linux12     231 Dec  7  2016 .bashrc
drwxrwxr-x.   2 linux12 linux12       6 Feb  2 19:37 Desktop
-rw-r--r--.   1 linux12 linux12     334 Sep 20  2017 .emacs
-rw-r--r--.   1 root   root         9 Feb  2 17:38 .magic_string.txt
drwxrwxr-x.   2 linux12 linux12       6 Feb  2 18:28 my_test
-rw-rw-r--.   1 linux12 linux12       0 Feb  2 18:29 textfile.txt
-rw-r--r--.   1 linux12 linux12     658 Aug  2  2017 .zshrc
[linux12@webminal.org~]$
```

rwx

In Linux, **rwx** stands for "read, write, execute."

d (directory) - Indicates this file is a directory

r (read) - The permission to allow reading of the file

w (write) - The permission to allow writing of the file

x (execute) - The permission to allow execution of the file

- means there are no permissions

The first entry on the left will indicate whether this is a file or a directory. Notice that Desktop is a directory while hello3.txt has a - in the first entry and therefore is a file. There are 3 sets of **rwx** . Reading from left to right, the first set of **rwx** is for the owner or User. The second set is for the Group and the last set is for everyone else.

So if we're looking at the hello12.txt file, we can see that the linux12 user can read and write and people in the **linux12** group can read and write. All others only have read access. No one has execute permissions. Since the file is a text, file, it isn't expected to be executable. However, this is a common issue with scripts - sometimes they don't work because they lack the execute permission. The solution is as simple as adding execute permissions.

In Linux, **rwx** is used to denote what permissions each user has for certain files. For example, in Ubuntu Linux, no one user has **rwx** permission for the root user's files so they need to use sudo to change to superuser mode to access these files.

chmod

This is used to change the access mode of a file and can be read as an abbreviation of "change mode."

The first argument after chmod selects whether user, group, others or all are going to change. The shorthand for this is u for users, g for groups, o for others, and a for all.

The second argument selects what is changing. Are you adding (+) or removing (-) permissions?

The third argument selects which of the 3 properties are changing. Are you changing read, write, or execute permissions?

Some people describe this as specifying who changes, what changes, and which is changing.

For example, suppose you create a script named bubba.sh but it doesn't run as shown below. Can you figure out why it doesn't run?

```
[linux12@webminal.org~]$ ls -l bubba.sh
-rw-rw-r--. 1 linux12 linux12 31 Feb 10 03:46 bubba.sh
[linux12@webminal.org~]$ cat bubba.sh
echo 'my best friend is bubba'
[linux12@webminal.org~]$
[linux12@webminal.org~]$ ./bubba.sh
-sh: ./bubba.sh: Permission denied
[linux12@webminal.org~]$
```

Yes, the problem is that it doesn't have execute permissions. So let's give execute permissions to ourselves and our group.

```
[linux12@webminal.org~]$ chmod ug+x bubba.sh
[linux12@webminal.org~]$ ls -l bubba.sh
-rwxrwxr--. 1 linux12 linux12 31 Feb 10 03:46 bubba.sh
[linux12@webminal.org~]$ ./bubba.sh
My best friend is bubba
[linux12@webminal.org~]$
```

To execute this command, you may need to have administrator permissions in webinal. However, you should have some level of execute permissions on most Linux platforms. Webinal locks their permissions down because it is a shared browser shell.

Exercise:

Change one of the document files you created for an earlier exercise to "read-only" using the chmod command. If you aren't sure how to do this or receive an error message, make at least three attempts and record your findings each time.

#

chown

The chown command is similar to the chmod command, and using this command allows you to change the file owner and group by entering commands at the command line. To use this command, type chown, the user you want to own the file then a colon followed by the group and the file itself. For example, you would type, chown root:root file2 if you wanted the root user and root group to own the file named file2.

```
[linux12@webminal.org~]$ chown root:root file2
chown: changing ownership of 'file2': Operation not permitted
[linux12@webminal.org~]$
```

In the example above, the user root and group root were not allowed but you get the idea.

Exercise:

Change the ownership of a list you created in a prior exercise, so you do not have permission to access it via the terminal. Write down the steps you took.

#

setuid

Programs and scripts usually run with the permissions of the user id and group id. The setuid command allows someone to set the user ID or assign its privileges to another user. For example, if you wanted a program with normal user access to be run with admin access, the setuid command would be used.

Be very careful when using this command because it can cause security issues by circumventing the natural segmentation that Linux provides us.

How can you tell when setuid has been set? Look for the s in the place the x should be in the user field.

```
[linux12@webminal.org~]$ls -l hello.bin
-rwsr-xr-x. 1 linux12 linux12 27 Feb  2 22:08 hello.bin
[linux12@webminal.org~]$
```

#

setgid

setgid can be read as "set group id," and is similar to the setuid label but for the group privileges and it covers both directories and files.

#

User Unique Directories

/home/${USER}

${HOME}

This is the home directory for a specific user. Since this is the user's home directory, the prompt will have a $. A quick way to get here is to type cd $HOME or cd ~

```
[linux12@webminal.org my_photos]$ pwd
/home/linux12/Desktop/my_photos
[linux12@webminal.org my_photos]$ cd $HOME
[linux12@webminal.org~]$ pwd
/home/linux12
[linux12@webminal.org~]$
```

/etc/passwd

This is not the directory where individual user passwords are kept; rather, it is where user accounts are defined. You can find the user's username and user account's ID number through this directory, and you can also find the account's group ID number here. In some cases, you can also access the user's full name, home directory, and the location of their default shell.

Passwords, along with other important user and system login information, are kept in /etc/shadow. Even though it might be tempting to try to edit some of these files manually, it's best not to. As mentioned earlier, Linux may store a program in many different places, and changing an individual file without changing the related files may result in files being out of sync and certain programs becoming damaged.

Processes and Jobs

In this section, we'll cover how to check the processes that are running, how to put processes that do not require user input into the background, and how to bring processes back to the foreground again. We'll also cover the difference between checking which processes are running and which jobs are running, and cover some different methods for killing jobs and processes.

A process is a single program that is executing. A job can consist of multiple processes.

#

ps

This command stands for "process status" and checks what processes are running. Using it is just as easy as typing any other command with options.

The ps command allows the user to view the status of processes currently running. To use this command, type ps into the terminal. You should get a printout that looks something like this:

```
[linux12@webminal.org~]$ ps
 PID TTY          TIME CMD
2531 pts/46   00:00:00 ps
19845 pts/46  00:00:00 sh
[linux12@webminal.org~]$
```

Let's go over each part we see on the chart individually. To the far left, you see the letters PID, which stands for "process ID." Next to it, are the letters TTY, which stands for "terminal type." Next to TTY is the TIME followed by CMD, which represents the commands that are running. Every program that executes has a process ID so in this case, we have 2 programs running.

Unfortunately, this isn't very useful. As you can see in the table above, this ps command is only showing us information about 1) itself and 2) the shell terminal that's open. This is okay if all we're doing is playing with the terminal and trying to learn the commands but we may need to add more options before this command becomes useful.

ps -elf

Instead of just settling for simply ps, we can add a few options to make it more useful for us. Adding -elf gives us a long listing of processes with a lot more detail. Let's go through the options that we've added to this command.

The -e stands for "extended." You can use this command in this case to view information that was not provided previously. Enter ps -e into your terminal and see what the results are. Can you understand the information given to you?

If not, try adding the -l option. The -l stands for "long listing." Type ps -el into the terminal. Now, compare it with just the table with the -e option. What differences do you notice?

Next, we will add the -f option, which stands for "force." Enter the entire command: ps -elf into your terminal. What's different about this table vs. the other two? Record your findings in your notebook.

```
[linux12@webminal.org ~]$ ps -elf
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S RajNave+   390 32694  0  80   0 - 29117 n_tty_ Jan27 pts/19   00:00:00 -sh
0 T RajNave+   428   390  0  80   0 - 38048 signal Jan27 pts/19   00:00:00 vim bold.txt
0 T RajNave+   982   390  0  80   0 - 38048 signal Jan27 pts/19   00:00:00 vim bold.txt
0 S LordAsh+  1007     1  0  80   0 - 28899 futex_ Feb08 ?        00:00:00 nano otro
0 T RajNave+  1966   390  0  80   0 - 38048 signal Jan30 pts/19   00:00:00 vim pot.ttx
0 T RajNave+  2204   390  0  80   0 - 38048 signal Jan30 pts/19   00:00:00 vim pot.ttx
0 S henri0   17765 17764  0  80   0 - 29049 n_tty_  2019 pts/80   00:00:00 /bin/sh
0 S beyondc+ 17791 17790  0  80   0 - 29049 wait   Jan17 pts/96   00:00:00 /bin/sh
0 S beyondc+ 18646 28937  0  80   0 - 29050 n_tty_ Jan17 pts/26   00:00:00 bash
0 S xkrater+ 19455 19454  0  80   0 - 29049 n_tty_  2019 pts/107  00:00:00 /bin/sh
4 S linux12  19845 19833  0  80   0 - 29116 wait   Feb08 pts/46   00:00:00 -sh
0 S rishabh+ 19880 19879  0  80   0 - 29049 n_tty_  2019 pts/91   00:00:00 /bin/sh
0 S penguwh+ 20136     1  0  80   0 - 40585 futex_  2019 ?        00:00:01 top
1 S RSmith2  31673     1  0  80   0 - 262516 read_e Jan23 ?       00:00:00 vmlinux ubda=/common_pool/.fsi
1 S RSmith2  31674     1  0  80   0 - 262516 unix_s Jan23 ?       00:00:00 vmlinux ubda=/common_pool/.fsi
[linux12@webminal.org ~]$
```

Exercise:

Open a few programs and use the ps command by itself without any added options. What did you get?

Then, try adding the -e, -l, and -f options. Were your results any different? Do you see where you would possibly apply this? Write down your findings and your answers in your notebook.

Answer this question: how can I use options with commands to create even better commands?

#

top

This is like the "task manager" command in Windows. It shows you the programs and processes. Type top in the terminal to use it. To get a list of commands you can use with top, type top and a list of commands should show up. It even allows you to kill processes on the top screen by pressing k.

The top command is used to show Linux processes. You can also combine it with other commands to put it to different uses. For example, you can see active processes for only user processes, and you can also type in **Shift+P** to sort processes according to their CPU usage. This makes it much easier to spot the processes that gobble up memory and find and kill stubborn, hung up processes.

You can also set the top command to close after refreshing itself a certain number of times by using the -n option.

#

sleep

This command allows you to pause the system or an individual script by creating a dummy process for it to work on. This can come in handy when you're programming or testing the system for weaknesses. To use this command, type sleep and the amount of time you want the system to pause into the terminal, followed by s, m, h, or d for seconds, minutes, hours, or days, respectively.

If you need to pause the system by initiating the sleep command, it's a good idea to not have the sleep command run for more than one full day. Very few tasks would require you to put the system on hold for longer.

The sleep process is a way of pausing a running processes or script to give other processes time to complete. This command has many uses in programming. For example, if a script is interfering with launching a certain program, you could elect to put that script on hold while running the other program. It might also be helpful in situations where you are testing new software or hunting for bugs or weaknesses.

Options:

Like other commands we've explored, the sleep command can be given with other commands, or even used as part of a mini-program with the use of the while loop or other bits of Bash programming. Once you've mastered the basics, you may think of all sorts of uses for it, depending on what you use Linux shell scripting for.

Exercise:

Use the echo command with the sleep command. Type "time (echo "string"; sleep <time>; echo "string") and set the time for only a few seconds. Record the results in your notebook.

#

jobs

In Linux, a job refers to a group of processes. Rather than having to enter commands for each individual process, you can, instead, enter commands that affect the full job, saving you time and making the process more efficient.

Use the jobs command to display the status of jobs started in the terminal window. For each session, jobs are numbered starting from the number one. Instead of using PIDs, some programs use job numbers; the foreground and background commands are examples of this.

Suspend a Job

To pause a job, type Ctrl + z. Notice the ^Z in the output below - that is the Ctrl + z.

```
[linux12@webminal.org~]$ sleep 30s
^Z
[1]+  Stopped(SIGTSTP)      sleep 30s
[linux12@webminal.org~]$ jobs
[1]+  Stopped(SIGTSTP)      sleep 30s
[linux12@webminal.org~]$
```

Note: there are times when you might want to use the ps command instead of the jobs command, like when you need to know what processes are running and eating up too much memory. The ps command gives more information than the jobs command.

Exercise:

As an exercise, let's start a couple of programs and suspend them.

#

Foreground vs. Background

Jobs typically run in the foreground. In other words, the job locks the terminal until it completes at which time you can start a new job. This can be inefficient if you have a job that takes a long time - you can do other stuff while that job is running. In another case, if you have a job that does not require user input, you have the option of moving it to the background. Since Linux is a multi-tasking operating system, it has the ability to do more than 1 task at a time.

Use & to Start Job in Background

You can also put the ampersand sign & at the end of the program to start the program in the background. Doing so would look something like:

```
$program &
```

To start a job as a background job, simply type a & at the end

```
[linux12@webminal.org~]$ sleep 10s &
[1] 26889
[linux12@webminal.org~]$ jobs
[1]+  Running                 sleep 10s &
[linux12@webminal.org~]$ jobs
[1]+  Done                    sleep 10s
[linux12@webminal.org~]$ jobs
[linux12@webminal.org~]$
```

In the example above, we started a job in the background and then checked to see what jobs were running.

Put Paused Job in Background

To move a foreground job to the background, first type **Ctrl+z** to pause the job and then a prompt will appear. At the prompt type bg and the number of the job you want to be moved to the background.

bg <job #>

```
[linux12@webminal.org~]$ sleep 30s
^Z
[1]+  Stopped(SIGTSTP)        sleep 30s
[linux12@webminal.org~]$ bg 1
[1] sleep 30s &
[linux12@webminal.org~]$ jobs
[1]+  Running                 sleep 30s &
[linux12@webminal.org~]$ jobs
[1]+  Running                 sleep 30s &
[linux12@webminal.org~]$ jobs
[1]+  Done                    sleep 30s
[linux12@webminal.org~]$
```

Notice how a job was started in the foreground above and then stopped with **Ctrl-z** before being put into the bg. Note how the # of the job was 1.

Move Background Job into Foreground

Use a similar method to move jobs from the background into the foregound.

fg <job #>

To move a job back into the foreground, you can use the fg command. Type the job ID number you want to move back into the foreground along with the command.

```
[linux12@webminal.org~]$ sleep 30s
^Z
[1]+  Stopped(SIGTSTP)      sleep 30s
[linux12@webminal.org~]$ bg 1
[1] sleep 30s &
[linux12@webminal.org~]$ jobs
[1]+  Running               sleep 30s &
[linux12@webminal.org~]$ fg 1
sleep 30s
[linux12@webminal.org~]$
```

#

How to Kill a Job

Killing a Job

You can use the kill command to kill a job. You can also use the kill command to stop or end a process. To stop a process, type the word kill and the listed ID number of the process you want to stop. For example, if you want to stop a process that is marked as "1572," type k 1572, and the process should stop. The exception is if the process is something related to the kernel or something you don't have the rights to alter.

Kill Job in Foreground with CTRL + C

You can also pause a job in the foreground by pressing the **Ctrl+c** keyboard

shortcut. This is the same as SIGINT or kill -2.

Remember that the kill command can be used in different levels of severity to stop programs, from "telling" it to stop itself to sending a signal to the program to terminate it. You can also use the kill command to restart a process.

Kill Job in Background

kill

What if you have a stubborn job or process that just won't quit? If you have a hung up program, you may want to try the kill command. The kill command is used to stop and end processes. In some cases, the kill command can be used with other instructions to resume processes as well.

To stop a process, use either ps or jobs to locate either the process ID or job ID number of the program or process you want to end, and then issue the kill command. Typing kill followed by the job ID number will cause a signal to be sent to the program to terminate. If the program refuses, you can type kill -9 and the process or job ID number. The process should terminate once you do.

You can also use the kill command to stop processes by typing kill -STOP and the process ID. If you would like, you can even resume a process by typing kill -CONT and the process ID number. This is another possible option if you do not want to use other options to stop and start processes.

Exercise:

Go and download a practice file, then kill the job in three different ways. Record your findings in your notebook. Refer to the previous paragraphs if you need to refresh yourself on the kill command.

Restarting, Stopping, or Removing Services

In Linux, individual programs can crash without taking the entire system offline. When this happens, you might want to restart, stop, or remove these services.

To do this, type in sudo /etc/init.d and the name of the service you want to restart, remove, or stop. Let's use Apache as an example, so a command to restart the service would look like:

```
$ sudo  /etc/init.d/Apache restart
```

Explanation:

Being able to restart only individual services prevents the need to reboot the whole operating system to clear small bugs in programs or receive needed updates. Not having to reboot a Linux machine saves time. Restarting the entire operating system could take minutes or more while restarting individual programs may only take a few seconds. This is helpful if you are using your machine to run web servers or perform other, more complicated tasks.

Chapter Summary

- In this chapter, we covered how the Linux security system works, the rights of the root user as compared to other users, how to change permissions of files, how to view the jobs and processes, and how to kill jobs.

What Did You Think of Linux Command Line: An Admin Beginners Guide?

First of all, thank you for purchasing this book **Linux Command Line** *. I know you could have picked any number of books to read, but you picked this book and for that I am extremely grateful.*

I hope that it added at value and quality to your everyday life. If so, it would be really nice if you could share this book with your friends and family by posting to Facebook and Twitter .

If you enjoyed this book and found some benefit in reading this, I'd like to hear from you and hope that you could take some time to post a review on Amazon. Your feedback and support will help this author to greatly improve his writing craft for future projects and make this book even better.

You can follow this link to https://www.Amazon.com/gp/customer-reviews/write-a-review.html?asin=B085C1RJX9 *now.*

I want you, the reader, to know that your review is very important and so, if you'd like to **leave a review** *, all you have to do is click* here *and*

away you go.

I wish you all the best in your continued learning of Linux and the Linux Command Line!

Thank you!

Troy

CHAPTER NINE: PROJECT IDEAS, MORE COMMANDS, AND CLOSING...

Project Ideas, More Commands, and Closing Thoughts

Quiz Time!

Close your notebook first and try to answer as many of the following questions as you can without looking at what you wrote down. If you are unable to answer certain questions, mark them with an "x" or a checkmark and calculate your score.

Once you finish giving yourself the quiz, you may look in your notebook and answer the questions you were previously unsure about.

Ready? Let's begin.

Linux Installation

1. When installing Linux, what kind of file should you download to make sure it installs correctly on your virtual machine?
2. True or false: Google Chrome is an operating system built on a kernel of Linux.
3. True or false: Arch Linux was created to offer Linux users the Linux experience without all the extra "bells and whistles" that could cause clutter and may ultimately be unneeded for certain projects.

Basic Files and Directories

4. What is the difference between the root folder and the root user?
5. Is there any danger when storing files in the root directory in Linux?
6. What does the lib library file contain?
7. Is capitalization important in Linux syntax?

Linux Shells

8. What is a shell? How is it different from a terminal?
9. What is a GUI interface? How is it different from a command line interface?
10. Is a desktop environment GUI or CLI?

11. What is the main difference between Bash and csh?
12. Do all Linux operating systems come with a desktop environment?

The Command Line

13. What does cd stand for? How is it used?
14. What's the difference between the absolute path and the relative path?
15. If you wanted to know the absolute path of your current working directory, which command would you use?
16. What does the pwd command stand for? What does it do?
17. Which command would you use if you wanted to know a file's type?
18. What are some features of the vim text editor?

Manipulating Files

19. How would you move a file from one directory to another?
20. How would you copy a file?
21. If you don't want a file on your system anymore, which command would you use to remove it?
22. How would you remove an empty directory?
23. How would you remove a full directory?
24. Name some ways to view the contents of files.
25. How would you create a blank file?
26. What does the cat command do?
27. Which command would I use if I wanted to print text to the screen?
28. How is the less command different from the more command?
29. If I wanted to view the last five entries of a list, which command could I use?
30. If I wanted to view the first five entries, which command could I use?

More Advanced Commands

31. Which command could I use if I wanted a list of all files with a certain title?
32. Which command could I use if I wanted to search for a file with a certain title?
33. The command grep stands for what? What is grep used for?
34. Which command could I use if I wanted to know how many words, lines, and bytes a document contains?

35. What are wildcards? What do they do when added to a command?

Files and Directories

36. What are two ways to compress a file in Linux?
37. Which commands could I use if I wanted to compare two files?
38. Which command could I use to display directories and subfolders?
39. How could I download a file from the command line without using sudo apt-get?

Other Commands

40. Which command could I use if I wanted to know if a website is responding?
41. Which command could I use to create my own shortcuts or custom terminal commands?

Help Commands

42. What are some advantages to using the man command over the --help option?

Understanding Linux Security

43. What is the sudo command, and how is it used?
44. How does a root user differ from other users?
45. What is one potential security problem about how Windows grants permission to programs? How does Linux solve this?

Files and Directories

46. What does rwx stand for? Why is having permissions for files and users important?

Users and Groups

47. What's the difference between chmod and chown?
48. How would you set the user ID?
49. How would you set the group ID?

Processes and Jobs

50. What's the difference between a process and a job? Give an example of each.
51. What are some potential advantages of the sleep command?

Foreground, Background, and Killing Jobs

52. How do you put a process that is already running into the background?
53. How do you tell a process to go to the background at the time you run it?
54. How do you kill a process?

Linux System and User Management

55. What does the top command do?
56. What does df do? How is it different from du?
57. How would you check to see how much disc space you have?
58. How would you check to see who is logged on to your network?
59. How would you find your hostname?
60. What are quotas, and who can edit them?
61. How would you add a user? How would you remove a user?
62. How would you change a user's details?

Passwords

63. What does the chage command do? How is it different from the passwd command?
64. What's an easy way to set a date of expiration for a password?

Groups

65. How would you add a group? How would you remove a group?

Now, tally up your score. Round to the nearest tenth if needed. How did you do?

If you got 100% of these questions correct, great job! You might know the basics of the Linux command line now, and you might be ready to take on more complex and advanced projects.

If you scored below 100%, make a note of all of the questions you missed and go back to the chapters where they are found and look them over. Keep practicing on the command line until you feel comfortable with all of the commands.

Project Ideas

Now that you have a general idea of the Linux file system and how it is structured, along with how to use its basic commands and options, you may want to further your learning by attempting some basic projects.

Basic-Beginner Friendly

Continue Bash

You might want to continue learning the Bash scripting language and try your hand at writing more complex commands. You may even try creating a labor-saving program or two. Such a project will likely require commands like alias and grep, combined with techniques like piping. After getting familiar with stringing basic commands together, you can start to use conditional checks and loops to really power up your scripts.

Let's pretend you want to write a program to find the text files in the current directory and limit the result to 2 files.

```
[linux12@webminal.org~]$ ll
total 32
-rwxrwxr-x. 1 linux12 linux12   31 Feb 10 03:46 bubba.sh
drwxrwxr-x. 3 linux12 linux12   22 Feb  2 21:50 Desktop
-rw-rw-r--. 1 linux12 linux12  324 Feb  2 23:36 hello12.txt
-rw-rw-r--. 1 linux12 linux12   81 Feb  2 22:14 hello3.txt
-rw-rw-r--. 1 linux12 linux12   27 Feb  2 22:08 hello.txt
-rw-rw-r--. 1 linux12 linux12 6084 Feb  2 22:30 longtext.txt
-rw-rw-r--. 1 linux12 linux12   17 Feb 11 03:13 mynewfile.txt
drwxrwxr-x. 2 linux12 linux12    6 Feb  2 18:28 my_test
-rw-rw-r--. 1 linux12 linux12    0 Feb  2 21:33 photos.png
-rw-rw-r--. 1 linux12 linux12   31 Feb  2 20:12 textfile.txt
[linux12@webminal.org~]$ find . -name "*.txt" | tail -2
./longtext.txt
./hello12.txt
[linux12@webminal.org~]$
```

You might be able to use simple commands like this when creating automated messages, such as for group emails, and there are many other things you could do with the commands you've already learned as well. Searching on the internet is a great way to get an idea of the things you could do. Type in something like "scripting basic programs in the Bash shell" and click on the links, tutorials, or videos that interest you. If you have a specific project in mind, tailor your search to your specific needs.

Give yourself some more practice with piping and using grep. Mix commands together or apply the commands you know to a project to expand your learning and get an idea of how it all fits together.

Learn the Basics of a Programming Language

Python

You may be surprised to find out that a lot of programs in the Linux environment

are created using the Python scripting language. Many distributions of Linux come with Python pre-installed, and distros of Linux that might have been made especially for aspiring programmers and hobbyists, such as Raspbian, may even have Python programming tutorials as standard desktop programs.

Improve your coding knowledge by taking the next step and learning how to write a simple program that can be run on your Linux machine.

Some suggestions for where to start may include just doing an internet search for "learn Python." You may find many free tutorials among paid options, or you can purchase or borrow an ebook, via Amazon's Kindle Unlimited service for example.

Feel confident about tackling your next project.

C and C++

To use other Linux features, you might want to learn some basics of the C programming language. Doing so will enable you to enter commands into the csh shell, as mentioned before. You may also learn how to write programs and apps for the Linux OS.

Set up a Raspberry Pi Project

Raspberry Pi is a simple mini-computer that may be small enough to fit into the palm of your hand. Don't let its size fool you; it can still be used to build amazing projects.

Simple All-in-One Computer

Have you ever wanted an "all-in-one" desktop computer (computers that come with the monitor and all of the hardware right out of the box and don't have to be connected to a tower), but thought they were prohibitively expensive? Good news! You can now create something fairly similar by setting up a Raspberry Pi mini-computer and hooking it up to an old LCD computer monitor.

For this project, you need to know how to:

- Assemble a basic computer.
- Download Linux onto a memory card.
- Configure options in the terminal.

This project is relatively easy to complete--merely put together a Raspberry Pi by use of the build kit that came with your Pi. If you need help, a quick internet

search should provide plenty of instructional videos.

Next, write a memory card with Raspbian if your build kit didn't already come with one. Insert the memory card that you wrote with the Raspbian operating system, then hook it up to power and boot it up.

Use a USB wand or dongle to connect the Pi to a monitor and anchor it in place. You now have a functional, "all-in-one" computer with no visible tower and wireless connectivity capability. Practice surfing the web and installing new programs on it. If you need expanded storage space, external hard drives are available and should connect with a standard USB port.

Intermediate

Set Up a Small Home Server

You don't have to wait until you learn a full coding language to do something productive in Linux. If you have a small computer that you aren't using and want to re-purpose, you could set up a home server for added security on your own network.

Before we get started, here are some of the commands that you might need to know:

- mkdir
- sudo apt-get install
- adduser
- addgroup
- hostname -i
- chmod

Step One: Select a computer to act as a server

Use a small computer for this project, either an old outdated computer or a Raspberry Pi mini-computer. You can increase the physical storage via external hard drives, and don't think you need a computer with a lot of RAM or processing power to act as your server.

Step Two: Install Linux

Use a USB copy of Debian or a version of Linux configured especially for server use. Put the USB copy of the program into the USB port of the computer you want to re-purpose as a server, and go through the prompts of the install. Select

the server version, then select the LAMP package install.

LAMP stands for "Linux, Apache, MySQL, PHP." Having these packages pre-installed during set-up will allow you to skip a few steps in the configuring process. You may elect to give your project a desktop environment, or you may decide that you don't need one and choose to keep it as only a command line.

Go through the installation process and follow the prompts to set everything up.

Step Three: Downloads

Open your command line and type:

```
$ sudo apt update

$ sudo apt install ssh

$ sudo ufw allow 22
```

Remember that sudo can be read to mean "super user do," and it may be a way of getting permission to perform administrative tasks that Linux would otherwise block you from doing. Try entering all of the commands separately rather than all at once.

Step Four: Enter the IP Address

Replace "ubuntu" with the name you want to use for your server, and enter your static IP address. If you don't know it, use the hostname command with the -i option.

That's it--you're now ready to turn this into a private server to store and remotely access home media files, such as pictures, music, videos, and more while keeping your data secure and free from the data-mining practices of cloud sharing sites. If you want, you can take a few additional steps and turn it into a web hosting server and have people pay you to host their websites.

Before you do this, you may want to learn one more command, however.

tasksel

<u>Explanation:</u>

This command stands for "task select" and is intended for those who want to use Ubuntu Linux for servers. If you know exactly what you want your server to do, you can just run the tasksel command to install the software you need to create a

specific server.

To use the tasksel command, type sudo tasksel. You may need this command once you set up your server.

Now, go make that server!

Advanced

The **Internet of Things** , or **IoT,** is fascinating. You can connect drones, wireless "smart" cameras, and even create small robotic appliances that send messages to a server to let you know their status. You could use IoT technology for many things, including vending machines that can sense when their inventory is low and can signal you to order more, eliminating the need for you to check them by hand. A popular use for this technology is the smart camera.

You can try making your own, small IoT project, like remotely connecting a Linux terminal to program a headless system to power a robot or remote-controlled camera. You might need to know Python and have a good idea of how to connect wirelessly to a Linux machine, as you probably don't want your new machine to have to receive commands via a terminal you have to access through a screen on the machine itself.

For this project, you might need to know how to:

- Program in Python.
- Understand basic robotics principles.
- Connect to a Linux system headlessly and wirelessly.

For more information on this, look up some drone and small robot builds. You may need special cables and some programming knowledge to get your creations to do what you want. Unfortunately, this book won't be covering much about programming languages outside of what you may need to know for basic Bash scripting. If you believe you're ready to tackle a larger project, don't let any of this discourage you--there are plenty of free resources on the web about getting started with the exciting world of AI. You may even find communities that will help you with your builds and projects, even providing step-by-step tutorials. There are various places to post questions for community feedback too, so you should definitely try it out if this interests you.

Suggestions for Further Reading

It depends on what you want to know. After reading this book, you should now

have a working knowledge of how to navigate the Linux terminal and enter simple commands. With the use of tutorials, you can create some pretty amazing builds.

If you want to get into programming, you will need to have a good grasp of a programming language like Python, and you need to get more comfortable entering commands into the Bash shell.

The exercises included in this book were very basic and not challenging at all, but the main focus was getting you used to the command line. To truly start programming in Linux, you need to then take the little bit that you've learned and expanded upon it. Look for resources and tutorials everywhere, and practice writing and executing small programs. Apply yourself like a dedicated student, and you will make amazing progress.

Remember, you've gotten all the foundations. Now, all you need to know how to do is put the building blocks together to create more impressive programs. Start out slow, then work your way up. Build small projects until you get comfortable with larger ones.

Closing Thoughts

I hope you've enjoyed this book, appreciate Linux a little more, and find the Linux command line less intimidating. If this book has helped you, consider leaving a five-star rating on Amazon and Goodreads, and don't forget to share this book with a friend who wants to learn Linux too.