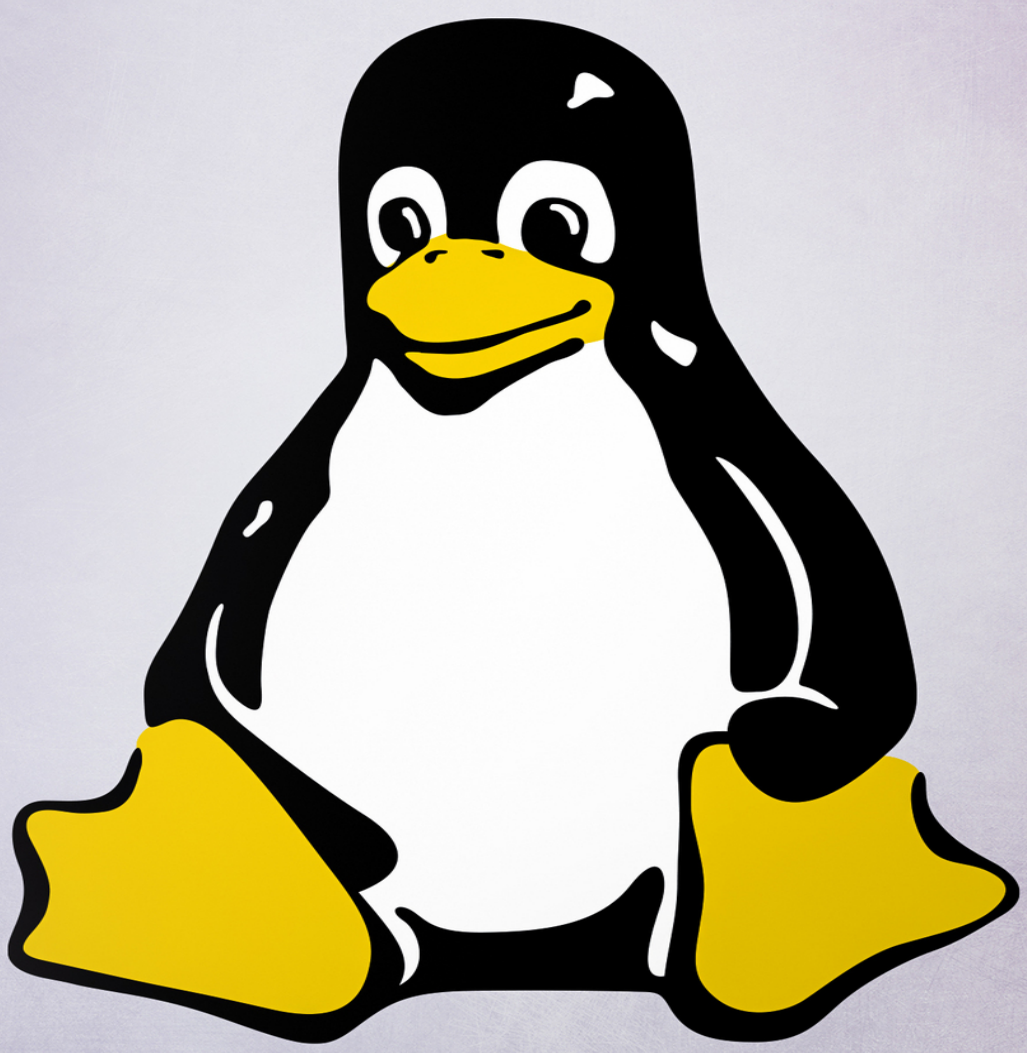
LINUX

FOR BEGINNERS

A Practical and Comprehensive Guide to Learn Linux Operating System and Master Linux Command Line. Contains Self-Evaluation Tests to Verify Your Learning Level

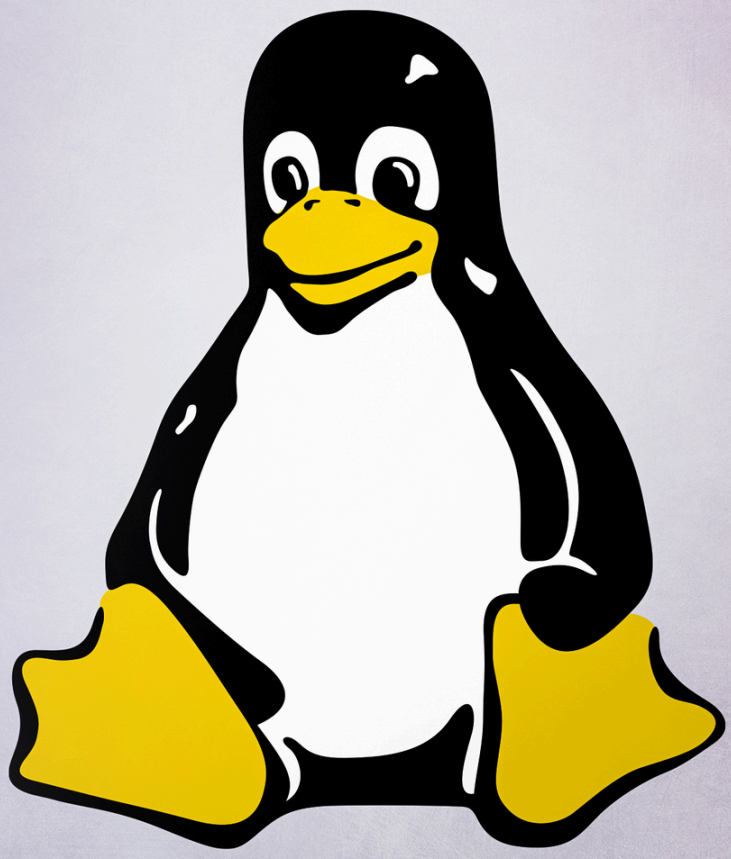


ETHEM MINING

LINUX

FOR BEGINNERS

A Practical and Comprehensive Guide to Learn Linux Operating System and Master Linux Command Line. Contains Self-Evaluation Tests to Verify Your Learning Level



ETHEM MINING

Linux for Beginners

A Practical and Comprehensive Guide to Learn Linux Operating System and Master Linux Command Line. Contains Self-Evaluation Tests to Verify Your Learning Level

By: Ethem Mining

Table of Contents

Introduction

If you have picked up this book, you are inevitably interested in Linux, at least to some degree. You may be interested in understanding the software, or debating whether it is right for you. However, especially as a beginner, it is easy to feel lost in a sea of information. How do you know what version of Linux to download? Or how to even go about downloading it, to begin with? Is Linux even right for you to begin with? All of those are valid questions, and luckily for you, *Linux for Beginners* is here to guide you through all of it.

Linux is an operating system, much like iOS and Windows. It can be used on laptops, large computer centers, on cell phones, and even smart fridges. If it can be programmed, Linux can almost definitely be installed, thanks to several features and benefits. Linux is small, secure, supported on other devices, and incredibly easy to customize. With Linux, you can create a setup that is exactly what you want, with privacy, security, and access to plenty of free to use software. This means that, once you develop the knowhow, you can create a customized experience that will do exactly what you need, allowing yourself to optimize the setup you have and ensure that the setup you have

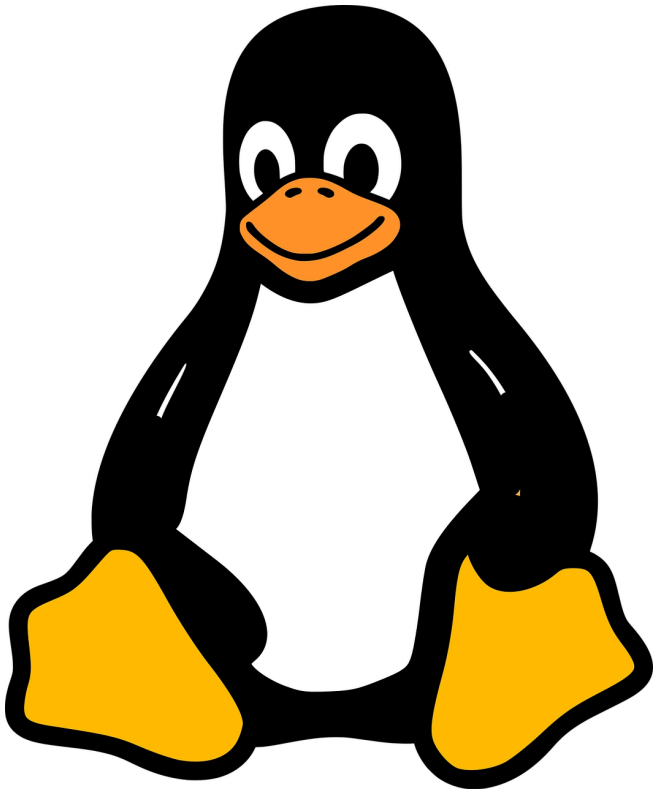
As you read through this book, you will be given a comprehensive guide to everything you need to know as a beginner to Linux. You will learn about why and how to determine which distribution of Linux is right for you. You will discover how to use the terminal, how to set up exactly what you will need on your system, and more.
When you are able to make your customized setup however you see fit, this means that you can make sure that you are always working within the constraints of the hardware that you are using. This means that older machines, which may struggle under a load of many modern operating systems such as Windows 10, can be optimized and used to their fullest potential without wasting valuable resources or processing power on aspects that are unnecessary, redundant, or even just detrimental to whatever it is that you need to do .

Ultimately, you will be provided with exactly what you need to know to get started with Linux, from start to finish. You will even be provided with several alternatives to Windows-specific applications that can be downloaded and used while running Linux on your device. Everything will be provided in the simplest terms possible, so you get a complete and thorough understanding of exactly what you need to know if you wish to get started with Linux, and at the end of each chapter, you will be given a short, five question quiz, as well as the answers to ensure that you are, in fact, comprehending the material that has been provided. Between receiving several step-by-step guides, questions, and lists of commands, you should have much of what you need to know to at least get started with the installation of your own distribution of Linux!

There are plenty of books on this subject on the market, thanks again for choosing this one! Every effort was made to ensure it is full of as much useful information as possible; please enjoy!

Chapter 1: What is Linux?

So, you have decided that you want to use Linux—or at least learn more about it than you already know. You are in the right place! Within this chapter, you will develop an understanding of what Linux is. First, you will discuss what operating systems are to develop the foundational information you will need to guide you through the rest of the book. From there, you will discuss Linux as an operating system, learning what it has to offer and why it is so commonly used by other people. From there, you will learn how Linux came to be. Lastly, you will begin to learn how Linux is composed, looking at the systems architecture, file hierarchy, and processes.



What is an Operating System?

An operating system, commonly abbreviated as OS, is the program that runs the hardware of a computer. In running the hardware, it also allows for the management and usage of the software. It also allows for the interaction

between the user and the hardware, facilitating acts such as input and output. It is essentially the intermediary, allowing the hardware to run the programs and the programs to make use of the hardware.

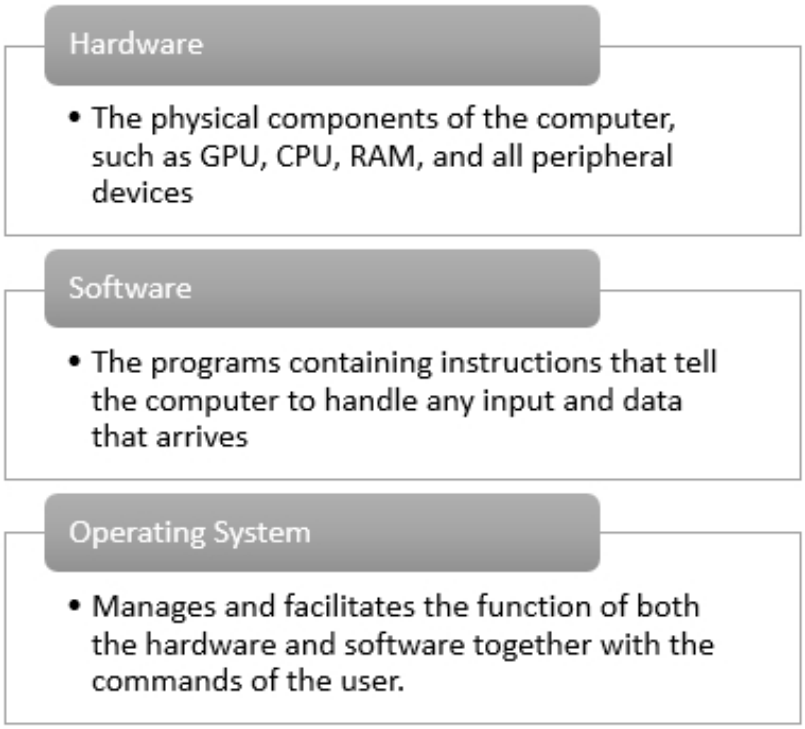
Of course, this means that the operating system is the single most important piece of software that you will ever run—without it, your computer will not be able to manage the hardware. Think of the OS as the brain of the computer. Just as your brain keeps your body running properly and reacts to interaction, so too does the OS keep the computer running properly and allow it to interact with the computer. When you use an operating system, you are essentially interacting with a program that allows you to speak to the computer, commanding it to do whatever you expect it to do without having to speak in a way that the computer understands. It is the translator, the intermediary, that allows you and the computer to interact, and without it, the computer is entirely useless.

The operating system's important job

As briefly touched upon, the operating system's job is to manage all software and hardware on the computer. Software refers to the programming on a computer—it is a list of instructions on how a computer should handle and process certain information in order to properly perform the tasks that you have commanded. It ranges from programs, such as running your internet browser or playing a video file all the way to the system running your computer. Each of these programs is what you have your computer to do, and the operating system makes sure it is run properly while also utilizing the hardware .

Hardware is the physical computer itself and all of the components that make it up. This includes the motherboard, the central processing unit

(CPU), RAM, graphics processing unit (GPU), hard drive, power supply, and all other components that you use, such as the monitor, mouse, and keyboard, and anything else hooked up to your computer to use it. Each of these computer components has their own specific purposes, which come together to allow for the processing of the computer software installed.



The operating system helps juggle all of those components together while also simultaneously managing the software and any user input, allowing for the computer to run. It ensures that each program that is functioning is able to access the proper hardware support necessary without infringing on the function of other programs at the same time. Essentially, the operating system manages it all to give you the smooth and seamless functioning you have come to expect from your technology through the following functions:

- **Command interpretation:** Allows for the translation of the commands given to the computer

- **Communication management:** Allows for the coordination and assigning of software resources
- **Device management:** Manages the usage of all devices
- **File management:** Manages any processes and activities related to files, such as organization, naming, retrieving, sharing, or securing
- **Input/Output management:** Translates and manages inputs and outputs
- **Job accounting:** Tracks the time needed and necessary resources by jobs and users
- **Memory management:** Allows for the allocation and de-allocation of memory that programs are in need of at any given moment
- **Networking:** Allows for processors that are not sharing any memory or hardware to communicate resources or data
- **Processor management:** Helps the operating system create and delete processes, facilitates synchronization, and allows for communication between processes that may be interacting with each other
- **Secondary storage management:** Makes sure that data is stored in the right storage to be accessed when needed
- **Security:** Protects the data stored within a computer system from any malware or unauthorized attempts to access it

Types of operating system

Given the wide range of functions for the operating system, it should come as no surprise that there are several different types of OS that all serve different purposes and come with their own strengths and weaknesses. This

section will provide a brief overview of the varying types of OS as well as the most common usage scenarios for them.

- **Batch operating system:** Designed to run jobs with similar needs that have been grouped together. The user is never directly interacting with the computer. Instead, the job is prepared offline and submitted to the computer operator to process. This is reserved for lengthy processes.

- **Multitasking/time-sharing operating systems:** This type of OS allows people at a different terminal (shell) to use the same computer at the same time. The CPU gets shared between the multiple users.

- **Real time OS:** These systems have minimal latency (lag between input/output) to allow for the response time to be nearly immediate. Ideal for military or space software systems that need to be able to react nearly instantly.

- **Network Operating System:** This OS runs on a server that is then accessible from several locations at once.
- **Distributed Operating System:** An extension of the network operating system, this type of OS makes use of several processors in several different machines, allowing for high-efficiency computing and processing.

- **Mobile OS:** This is any OS that is designed specifically for mobile devices, such as smartphones and tablets. The most commonly used mobile operating systems include Android and iOS.

Commonly used operating systems

Most of the time, any computer you buy will come with some sort of operating system already installed and ready to go. This means that if you

want to use most products right out of the box, you can choose to do so. However, you can also upgrade them or change out your operating system for another one if you find that another one. For the most part, the choice of operating system is primarily a matter of preference, though some OS will come with different software compatibility.

Perhaps the starkest difference between operating systems will be the graphical user interface (GUI- pronounced as gooey). The GUI is what you see on your display when you are using your computer and allows you to interact with your software, and in turn, hardware. This will be the combination of graphics and text that you see on your screen. While the GUI may be different between OS, they will all bring with them the same basic features and functions.
Nevertheless, if you have a preference for one type of OS over the other, that is a valid matter of opinion, and the one you choose should be the OS that works best for you. Nevertheless, take a moment to familiarize yourself with the three most common operating systems you will encounter from day to day: Microsoft Windows, macOS, and Linux.

- **Microsoft Windows:** Created in the mid-1980s, this is the most popular OS you will encounter in varying versions. The most recent is Windows 10, which was released in 2015. Nearly any personal computer (PC) that you will buy comes preinstalled with Windows, with the most notable exception being Apple products. It is fairly versatile in what the user can do, and is constantly getting updates to keep it secure.

- **macOS:** Preloaded on Macintosh computers, macOS is much less popular. It is calculated that less than 10% of the global operating systems are utilizing macOS. It looks sleeker but is typically only works with proprietary software and peripherals. Altogether, between the three operating systems, macOS is going to be the most expensive to run to its fullest capabilities. Because of this, macOS is a bit more on the restrictive end in usage and customizability, with many common programs, games, and other software not being macOS compatible.

However, the macOS tends to be on the less-hacked side, making it perhaps a bit more secure.

- **Linux:** Linux is a bit different than the other two—unlike the proprietary Windows and macOS, Linux is open-source. This means that it can be modified by anyone and distributed around the world for free. This makes Linux the least restrictive of the three major operating systems you will encounter, and yet it makes up less than 2% of the global operating systems in use. Despite this statistic, most servers will utilize Linux thanks to the versatility and customizability.

Linux as an Operating System

Linux, then, is an OS that is open-source—this means that it is readily customized and altered. Open-source operating systems are free to run, free to alter, and free to distribute, whether you have altered it or not. This means, then, that there are many different distributions of Linux floating around out there, as each person who has ever made a change or customized version for themselves is free to redistribute their own version. However, these distributions will be discussed in depth in Chapter 2.

Linux can be installed on a wide range of hardware, allowing for the development of software and running of applications. This OS was developed in the C programming language, and today, C is still the primary language used. Initially designed to be similar to UNIX, which will be discussed in the next subchapter, Linux has evolved far beyond its initial scope. It is now used on phones to supercomputers and everything in between. When you use Linux as an OS, you are utilizing the Linux kernel to manage the hardware, and then building up from that kernel to develop the rest of the OS with software .
The operating system that you think of when you hear Linux is actually an assortment of several different pieces. These pieces all work in tandem to allow the software programmed to function. These core pieces are the bootloader, the kernel, the init system, daemons, the graphical server, the

desktop environment, and the applications that you install. These pieces all serve various functions that will be crucial to your ability to actually run the OS in the first place.

- **Bootloader:** This is the software that will boot the computer—this means that it will make trigger the operating system to start running. Most of the time, there will be a quick splash screen that will display, letting you know that the OS is booting up. The splash screen will disappear as soon as the OS is ready to function, and that time is largely dependent upon the hardware that is running it. There are several different bootloaders, such as LILO (Linux Loader), LOADLIN (Load Linux), or GRUB (Grand Unified Bootloader).

- **Kernel:** This is where the actual name "Linux" came from. The Linux Kernel acts as the heart of the OS, allowing for the management of and interaction with the CPU, RAM, and all peripheral devices that are being used. It is the foundation for the rest of the OS, and without this kernel, you cannot run the device.

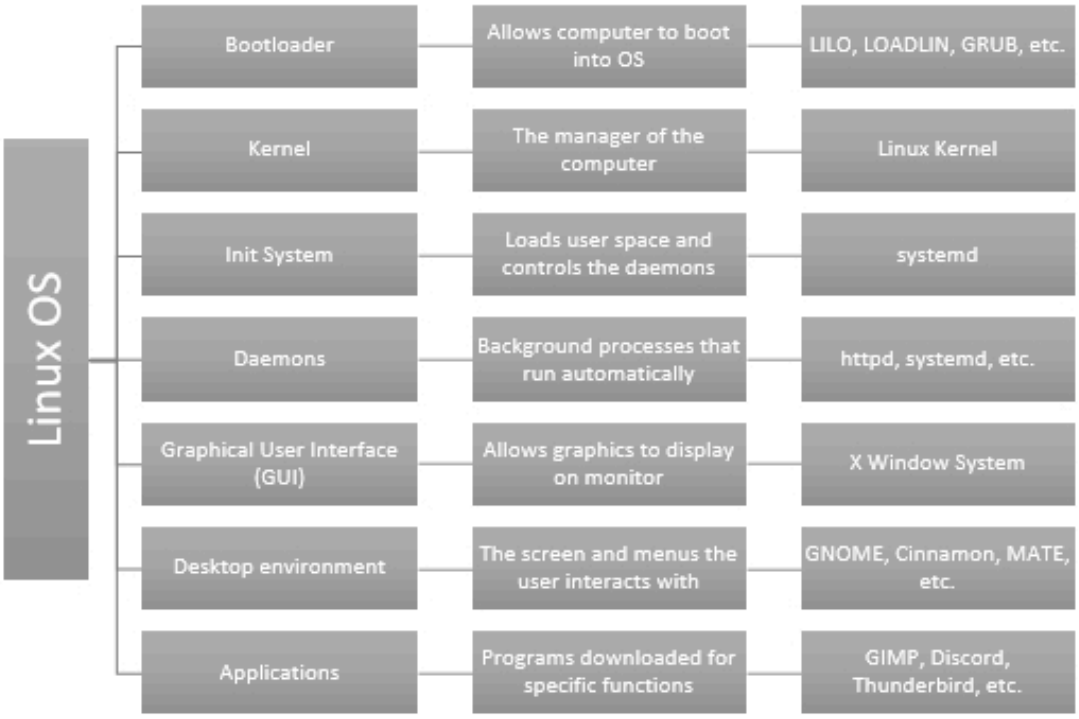
- **Init system:** This allows for the bootstrap, or activation, of the user space. Most often, you will see systemd as the init system.

- **Daemons:** These run several different background services that typically start on their own during the booting process or after you have logged onto the desktop. Think of daemons as background processes, and most often, you will see them in code with a "d" at the end, such as httpd as the web server daemon.

- **Graphical user interface (GUI):** This is a sub-system responsible for translating the computer data into the graphics displayed on your monitor of choice. A regularly used GUI in

Linux is the X Window System, more commonly known as X.

- **Desktop environment:** This is the part of the graphics that you are actually able to interact with. There are several that are readily available depending on your own preference, with some of the more common ones being GNOME, Cinnamon, MATE, Unity, etc.
- **Applications:** These are just like applications on your phone. This is any sort of program or software downloaded to serve a specific purpose. There is nearly an endless supply of different applications that can be downloaded and utilized with Linux, including many that are readily available on the other operating systems as well, such as GIMP for image editing, Discord for chatting, Thunderbird for emails, and more.
  Chapter 8 will give you a rundown of several alternatives to common windows applications for more examples of common Linux apps available to you.



From Unix to Linux

What is now known as Linux began first with the development of Unix? Developed from the Multics project from the Bell Laboratories Computer Sciences Research Center, the goal was to create a multi-user OS that would have single-level storage, dynamic linking, and a hierarchical file system. However, this project was discontinued in 1969.

Being discontinued did not deter a group of researchers, including Ken Thompson and Dennis Ritchie, to complete their project. Instead of Multics, however, they utilized the C programming language to rewrite their entire system, maintaining those core assets of Multics.
In the end, they created Unix. The programming language made this OS unique—it was portable. This meant that it could be moved off of the hardware it was on, allowing data to live long past the hardware's life span.

This newly created Unix system was developed further and allowed to be adopted commercially. It eventually spread further within academia, with the creation of Berkeley Software Distribution (BSD). This was further developed into NeXTStep, which eventually grew to become the foundation for macOS, and MINIX, designed to be educational and eventually became the reference for Linux Torvalds to create Linux.

Unix and its successors primarily remained locked behind licensures, and many different developers began to work to create free alternatives. Among these were Richard Stallman, a researcher who worked for MIT at the time. He started work on what became known as GNU, which was distributed as open-source software beginning in 1985.

Torvalds, frustrated with the licensure for MINIX, began work on his own OS in 1991. It was reminiscent of MINIX, with beginning development done on MINIX with the use of the GNU C compiler. However, over time, it transformed into its own project with its own developers to separate it out, and in 1994, version 1.0 of the Linux kernel was released .

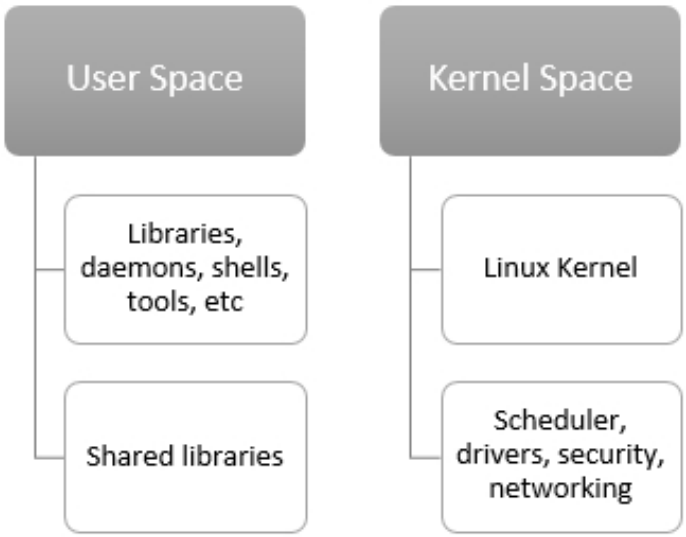
This means that there were two major influences from Unix to Linux—GNU, which remains a prominent component of many different distributions of Linux, and the initial development of Linux done within the MINIX system.
Most commonly today, when you hear “Linux,” you are most likely discussing the presence of both the Linux kernel and GNU. However, there are some systems, such as those on handheld devices, frequently use the Linux kernel with next to no GNU influence .

Systems Architecture of Linux

At this point, you have already seen a brief overview of the architecture of Linux—this is the breakdown of Linux that you saw in the chart earlier in the chapter. However, there are some details that need to be discussed in further depth. Within this section, you will learn more about the separation of user vs. kernel space, function of the Linux kernel, and the command line interface (CLI) known as a shell.

User space vs. kernel space

Wouldn't it be frustrating if you were working on your computer, having spent hours trying to solidify something that you were trying to accomplish, only to have the entire system shut down because it accidentally took up too much space for the OS to keep operating? Most people would agree that yes, it would be incredibly frustrating to have a program that would crash randomly due to running out of memory, and for that very reason, there is a special space dedicated solely to the OS on your memory in order to run it.
Your OS will break your memory down into two categories, separating them out entirely and completely restricting them. It creates a kernel space, which is designed to have enough space for the system to run in order to ensure that you never run out of space necessary to run the OS in the first place.
It also creates an extra safeguard, protecting your OS from any tampering from outside sources that may have been downloaded or initiated within the user space. By creating that sort of divide, the kernel is protected, and your computer should function smoothly.

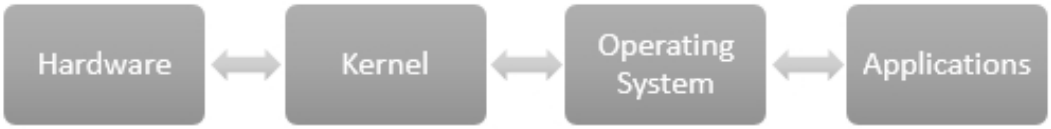


Every application must make system calls to the kernel within the kernel space if they wish to access system resources, such as making use of memory or network devices since the kernel is what tells the hardware to process in the first place .

The Linux Kernel

The Linux kernel is literally the namesake of the program itself and is crucial to understand. The kernel, as summed up earlier in the chapter, is the part of the operating system that controls the computer. The kernel acts as the mediator between the hardware and the operating system itself, allowing the OS to run applications in the first place. It schedules the order that applications will be run, accesses data, and even manages how memory is used within the computer.

This is the initial part of the OS that is booted upon startup, and the kernel contains the most important parts of the OS code.



As depicted in this graphic above, you can see the relationship between the processes clearly—this is a more in-depth version of the graphic introduced in the beginning of the chapter, showing you the step between the OS and the hardware now that you have a better understanding of how the OS will work in the first place .

Command line interface (CLI)

Most often, Linux is administered from what is known as a command line interface (CLI). This is also commonly referred to as a shell. Chapter 4 will be dedicated to discussing shells in more depth, but you should have a general idea of what they are before continuing on. The shell is the program that allows you to input commands from a keyboard. Within Linux, the most commonly used program is bash (bourne again shell), which is an enhanced version of the original shell program used by Unix that was written by Steve Bourne.

However, you cannot access the shell without another program first that will enable you to communicate with the shell in the first place. This is

known as your terminal emulator, and when you are able to install this, you are able to begin to interact with the terminal.
Opening the terminal emulator will open up another window that will allow for the interaction via typed commands. Like the shells, these commands and instructions on how to start and use a terminal will come later in the book. At this point, it is enough for you to know that interacting with the OS directly is an option for you .

File Hierarchy Structure of Linux

In order to function as effectively as possible, Linux makes use of what is known as the Linux File Hierarchy Structure (FHS). This defines the directory structure and contents within Unix-like operating systems, such as Linux. This particular hierarchy is managed, maintain, and regulated thanks to the upkeep performed by the Linux Foundation.

When using the FHS, all files and directories can be found under the root directory /, even if they happen to be stored elsewhere physically or virtually. However, some directories exist only under certain circumstances, requiring a particular system with specific subsystems installed. Most directories that will be discussed in this section can be found in all Unix operating systems, and they are commonly utilized in the same way, though the descriptions you will receive in this subchapter should only be considered true for Linux .

/ (Root)
This is the primary root of the entire file system. Everything can be linked back to root. Only the root user will have permission to write or edit under /, and you will see /root as the root user’s home directory. Note that /root is different from /.

/bin

This is a list of essential command binaries that must be available within single user mode. This will include binary executables, meaning files that are stored in binary format, making them readable to the computer, but not

to humans. You will also find other common Linux commands for single-user modes within this directory.

/boot

These are your bootloader files. You will see files such as:

- **Kernel initrd:** initial RAM disk—allows for the initial ability for the boot loader to load a RAM disk.
- **Vmlinux:** ELF (executable and linkable format) based file that is the uncompressed version of the kernel image for debugging.
- **GRUB file:** The bootloader package.

/dev

These are essential device files. These may include terminal devices, devices attached to the system, or USB.

/etc

These files are host-specific but still system-wide configuration files. You will find all of your programs' various configuration files within this section, as well as the startup and shutdown shell scripts that will be necessary to either start or end your programs.

/home

This is where you will find personal settings, your files that you have saved, and home directories. If you have downloaded and saved it, it will be here. It will most likely save with your profile or computer name in a format like /home/NAMEHERE.

/lib

This is the home to any libraries that are necessary for the binary files that you have saved in /bin/ and /sbin/.

/media
This is where you will find mount points for removable media. Most often, you will see the name of the format of the mount here, such as /media/cdrom if you have insterted a CD, or /media/usb-drive for a USB drive.

/mnt
This refers to temporarily mounted files. In this directory, temporary filesystems can be mounted by sysadmins.

/opt
This refers to optional application software. These files will house add-on applications, and they should fall under /opt, or a subdirectory somewhere within /opt .

/proc
A virtual system that houses process and kernel information. Typically, this is automatically generated by the system, with information about the current running process.

/sbin
This is another file for binary executables, but /sbin files are typically used for maintenance purposes by the system admin.

/srv
Standing for “service,” /srv holds data that is related to server-specific functions or services.

/tmp
These are temporary files that are most likely not going to be preserved when the system eventually shuts down and reboots. Typically, these are also restricted in the size they can take up.

/usr
This is the secondary hierarchy level for read-only user data. This will house most of the user utilities and applications. It is filled with several different files, such as:

- **/usr/bin:** binary files for user programs
- **/usr/sbin:** the system admins' binary files
- **/usr/lib:** libraries for /usr/bin or /usr/sbin
- **/usr/local:** Programs that have been installed from their sources
- **/usr/src:** the Linux kernel sources

The Process Architecture of Linux

Process architecture is the design that dictates the hierarchy of processes and systems when transforming any inputs to outputs in computing. While this sounds complicated to understand at a glance, hopefully, this subchapter will help you get through this process with ease.

The purpose of the process architecture

Within the Linux kernel are several subsystems, and the most important is the process scheduler. The purpose of the process scheduler is to control any access to the CPU, regardless of whether the access is via user process or from another kernel subsystem. This is all done by the process scheduler, which can itself be divided into four distinct modules as it interacts with the rest of the computer.

Linux processes

Commands issued to Linux trigger processes. When you insert a command into Linux, then, a new process begins. This process is assigned a 5-digit ID that allows the OS to track the processes—this will be known as the pid (process ID). While pids may be recycled, no two processes will share the same pid at the same time due to the pids being the process through which Linux is able to track the process in the first place.

This process is run one of two ways: In the foreground or the background. When in the foreground, where processes start by default, you are able to

insert your input to the keyboard, and you get output on the screen.
However, while these processes are running in the foreground, you are unable to complete other processes as well. For this reason, sometimes, processes get booted to background processes, where they run in the background, with no access to keyboard input, until that input becomes required. With it running in the background, other processes can be handled at the same time, meaning you can multitask until you are required to input further data.

The process runs as one of three distinct types: Parent and child, zombie and orphan, or daemon processes. Each of these different process types function slightly differently.

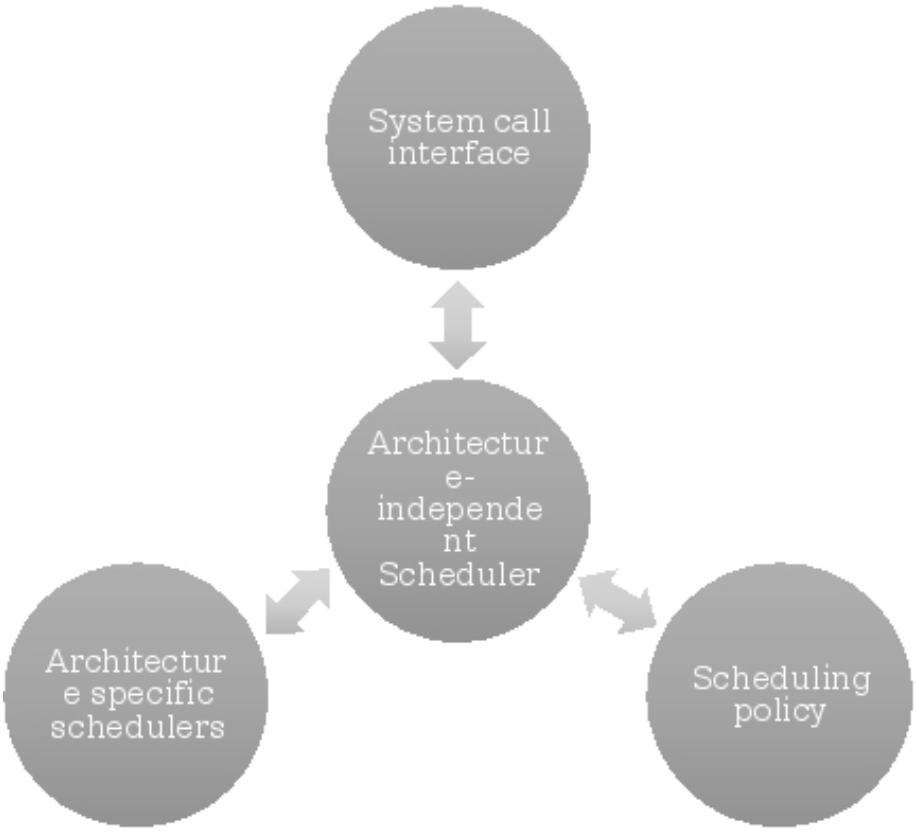
- **Parent and child process:** This occurs when a process creates another know-how. The created process is known as the child process while the parent process is the one that triggered the second.
- **Zombie and orphan process:** Upon finishing its function, the child process becomes terminated, and SIGCHLD, the signal from the interface to the parent process, informs the parent process of the termination of the child process. However, sometimes, the parent process is terminated before the child process terminates, creating a situation in which the child process is orphaned. The parent process still has an entry to the process table but is not functioning
- **Daemon process:** These processes are system-related and run in the background, most often running with permissions of root and waiting for processes that will work with them.

Process scheduler modules

The process schedule comes in four modules that will work together to organize the processes. These modules are:

- **The scheduling policy module:** This module judges which processes are granted access to the CPU, granting fair access to CPU to all processes.
- **The architecture-specific modules:** This is actually several modules—they are designed to share an interface that allows for the abstraction of details of any computer architecture. Their purpose is to communicate with the CPU in order to suspend or resume processes through knowing what information is needed to be preserved for processes and executing assembly codes that directly interact with whether a process with be suspended or resumed.
- **The architecture-independent module:** This module is the center of it all—it communicates first with the policy module to decide which process goes next, then tasks the architecture-specific schedulers to resume the right process. It then interacts with the memory manager to make sure that there is enough memory hardware available for the process to run effectively.
- **The system call interface module:** This module allows user processes to access only resources that are intentionally and explicitly exported by the kernel. This regulates the divide between user and kernel memory.

As pictured in the graphic below, the architecture-independent module is at the heart of this entire process and directly interacts with each of the other modules, allowing the entire process to run together effectively in a way that actually works. The scheduler essentially manages it all, with an entry for each process. The entire graphic encompasses what is known as the process scheduler.


System call interface
Architecture-independent Scheduler
Architecture specific schedulers
Scheduling policy

Chapter 1 Quiz

Congratulations! You have made it through Chapter 1. Try to answer these questions to ensure you understand the basics before moving on. The answer key will be on the page directly after this quiz.

1. **What is the operating system?**
    - **a.** The person or people operating the system
    - **b.** The program that controls the hardware
    - **c.** The program that controls the software
    - **d.** The program that controls the hardware and software
    - **e.** All of the above
    - **f.** None of the above

2. **Why do you need to keep the kernel space separate from the user space?**
    - **a.** Because you don't want the user space to run out of memory
    - **b.** Because you don't want to run out of space for the operating system
    - **c.** Because you don't want the kernel space to accidentally damage files on the user face
    - **d.** Because you don't want to mix personal and business files
    - **e.** All of the above
    - **f.** None of the above

3. **True or false: Unix came after Linux**

4. **True or false: Linux was built with Python**

5. **How many times can a pid be reused?**

a. Never—it can only be used once

b. A handful of times—so long as there are not two processes sharing the pid at one time

c. Infinitely, so long as there are not two processes sharing the pid at one time

d. All processes originating on one device share the same pid

Chapter 1 Answer Key

1. **D**. The operating system allows the user to control and interact with both the hardware and the software, but it is not the user him- or herself.
2. **B.** If the kernel were to run out of space for necessary functions, the system would crash, so keeping the memory split between kernel and user guarantees that there is always enough space for the kernel to function.
3. **False.** Linux was heavily influenced by Unix and Unix-influenced programs such as MINIX and GNU.
4. **False.** Linux was coded with C.
5. **C.** There is no limit to the number of times a pid can be reused, so long as there is never a current duplicate.

Chapter 2: Choose Your Distribution

So, you've made it to Chapter 2, congratulations! You have gotten through the background information that you will need to understand in order to proceed and have decided that you do, in fact, want to use Linux as your OS. Unfortunately, it is not as simple as just deciding to use Linux. Remember, Linux itself refers to the Linux kernel—the core of the operating system. From there, there are several other components, and while the core remains constant, your decisions on all of the other aspects of the OS, such as the GNU, the X server, and more, can all drastically alter your experience.

Some of the versions of Linux, known as Linux distributions, or more commonly shortened to "Linux distros," will be easy enough to hop right in to using, while others could be your worst nightmare immediately upon installing it, such as Gentoo, which requires all programs, including the kernel, to be built from source.

This chapter will guide you through understanding what a Linux distribution is, how to decide which distribution will be right for you and your usage, and then provide you with a brief overview of ten of the most common distributions you can find .

What is a Linux Distribution?

When you think of other operating systems, you would most likely think of Windows or macOS—two of the most commonly known and used operating systems out there. These are built from the ground up by a single company with incremental upgrades and changes, and because of that, you get distinct versions. You have Windows XP, Windows 7, Windows, 8, Windows 10, etc. with the latest version of Windows (Windows 10 at the time of writing this book), being the most up-to-date and supported. If you want to go out and buy Windows for your computer right now through Microsoft, your choice is to get Windows 10. Of course, you can get other, older editions through other sources, but they are not actively supported, nor are they going to always be updated for security. At a certain point, companies stop rolling out support for their older, more obsolete software.

Unlike Windows or macOS, however, Linux is not created by a single company. Linux can be mixed and matched.
So long as you have all of the right components together to create a complete OS, you can put it together. This means that you can, in fact, create an entire OS more or less from scratch, using the Linux kernel as your base, though that would be quite time-consuming, and you would have to put in some work to configure everything to work together. Alternatively, you can seek out Linux distributions.
The Linux distribution is essentially everything all neatly packaged together for you. It takes all of the open-source information you would otherwise need to seek out on your own and puts it together into a single OS, not unlike what you would receive when purchasing licenses to use Windows or macOS. They decide which desktops you will have, which programs will be loaded up by default, and more, typically with a layer of their own custom interface.

Of course, this also implies that there will be a nearly infinite number of different distributions available to you, all of which can provide all sorts of

different features. This is why it is important for you to do your research before choosing one .

How to Choose the Right Distribution for You

Now comes the hard part—figuring out which distribution will be right for you. As you read through the rest of this chapter, consider the questions that will be asked here. What are you installing the OS on? What is the purpose of the OS? Do you have any experience? What kind of software availability to you want or need? What kind of desktop interface do you want? How often do you need updates?
All of these are incredibly important questions to keep in mind. If you are not careful, you could end up with a version of Linux that you do not actually want, or cannot actually make use of.

As you read through each of the ten distributions that are provided to you, you will be provided with a brief overview of the OS itself, whether it is suited for beginners or those with plenty of experience, how regularly it is updated, and how compatible it is. You will need to choose one that works best for you .

Ubuntu

The first distribution we will consider is Ubuntu. Launched in 2004, it is now officially available in three different versions: Desktop, Server, and Core. Each of these editions can be run on a computer or in a virtual machine, making this a popular choice in OS for cloud computing, especially thanks to its support for OpenStack. In fact, Linux is the most popular desktop Linux distribution currently available.
Originally based on Debian sid, commonly referred to as the unstable distribution, thanks to its lack of stability. While Debian sid was unstable, Ubuntu has been built intentionally, avoiding the mistakes made within sid or other similar projects. In doing so, it created a web-based infrastructure, bug-reporting, and professional creation.

This particular form of Linux utilizes GNOME, Firefox, and LibreOffice by default, all updated, and it utilizes regular, predictable support updates. Ultimately, the creation was a well-formed OS that also includes a migration assistant for those transferring from Windows, support for latest tech, and it is compatible with ATI and NVIDIA graphics cards .

Difficulty rating: Good for beginners

Long-term support: Regularly updated distributions every 6 months with long term support releases as well.

Pros:

- Regular updates on fixed cycle and support periods
- Long-term support variants come with 5 years of updates for security
- Beginner-friendly
- Widely-used, meaning there is plenty of advice, opinions, and support available online

Cons

- Incompatible with Debian
- Regular updates often come with major changes that may be annoying to some
- Variants that are not LTS only have 9 months of security support

Linux Mint

Linux Mint is a distribution that is based on Ubuntu. This distribution was first launched in 2006, just two years after the release of Ubuntu. This particular form was created by Clement Lefebvre, a French-born IT specialist who resided in Ireland at the time of creation.
He had originally been maintaining a website that provided help and guides to people new to Linux, and during his time maintaining this site, he came up with an idea to develop his own Linux distribution in order to address several drawbacks present with other mainstream versions of Linux.

Though commonly referred to as "Ubuntu done right," Linux Mint is more than just Ubuntu with a refresh—it includes several tools designed to enhance the Linux Mint experience, often aptly named with "mint," such as mintDesktop, mintMenu, mintUpdate, and more. The mint tools are designed to be more usable, and greatly increase the ease of use, making this one of the easiest forms of Linux to pick up thanks to the emphasis on user-friendliness. One of the biggest draws to Linux Mint is the fact that the developers are always ready and happy to implement good suggestions that will help better the program.

Difficulty rating: Good for beginners

Long-term support: Not on a fixed schedule, but is regularly updated, especially shortly after each of Ubuntu's LTS releases.

Pros:

- mint tools designed for the OS
- User-friendly
- Includes multimedia codecs

- Open to the requests and suggestions of the users

Cons

- No security advisories
- Not predictably updated

MX Linux

MX Linux has a bit of a history to it. Originally known as MEPIS Linux, a Debian-based distribution meant for both personal and professional purpose desktops, featured several cutting-edge features at the time of its creation. However, despite this cutting-edge availability, MEPIS Linux eventually was subject to discontinuation. Its users, however, were not to be deterred. Merging MEPIS with another Debian-based distribution known as antiX resulted in what is known now as MX Linux.

This is based on Debian's stable branch, with primary components from both the antiX and MEPIS communities. In particular, it utilizes the Xfce desktop along with a vertical panel on the side instead of the common horizontal control panel across the bottom of most computers. Overall, this distribution is recognized as one that offers plenty of modern apps that are occasionally updated, along with a stable base, allowing for the creation of good, reliable performance without losing those features that people want most .

In particular, MX Linux is recognized for its MX-Tools—a series of graphical administration utilities that allow for ready access to several functions, such as managing the user accounts on that system, installing codecs, and utilizing and installing software packages.

Difficulty rating: Good for beginners

Long-term support: Features occasional updates through backports

Pros:

- Immediate support for graphics drivers, codecs, and browser plugins allow for near-instant startup out of the box
- Stable
- Updated periodically
- Convenient features

Cons

- Simple and dated appearance
- The installer and other configuration tools may take some time to adjust to

openSUSE

Dating back to 1992, openSUSE came about when four German Linux enthusiasts launched it. Thomas Fehr, Roland Dyroff, Hubert Mantel, and Burchard Steinbild started out selling floppy disks with the German edition of Slackware but quickly created their own program: SuSE (Software und System Entwicklung, or software and system development). In 1996, SuSE Linux became its own distribution, and over time, more and more features were adopted, such as the RPM package management tool, a graphical system administration tool designed for usability, and more.
They released frequently, kept fantastic documentation, and made their product readily available, and SuSE Linux grew more and more popular.

In 2003, it was acquired by Novell, INC, and eventually transferred to Attachmate in 2010. These exchanges in ownership led to major changes in development, and eventually, the change in name from SuSE to openSUSE. Despite the constant changes, however, it is readily available for free. You can find openSUSE in two primary editions. Leap grants users a platform that is stable and has several years of support. Tumbleweed, on the other hand, uses a rolling release environment, and this is made smooth by the easy configuration and automated filesystem snapshots and boot environments.
Difficulty rating: Good for beginners with some experience

Long-term support: Plenty of support offered

Pros

- Configuration tool is comprehensive
- Plenty of software packages
- Fantastic website infrastructure
- Plenty of printed documentation and support
- Btrfs with boot environments

Cons

- Bloated and heavy desktop setup
- Graphical utilities are slow

Fedora

Fedora itself was not announced and revealed until 2004, though its roots back to 1995 with Red Hat Linux, which was launched by Bob Young and Marc Ewing. Their product was released and updated several times, and in 1997, it added the RPM package management system, along with several other advanced features, which quickly shot Red Hat to popularity, eventually even surpassing Slackware Linux as the most widely-used distribution at the time.

In 2003, after Red Hat Linux 9 was released, the company made some significant changes, keeping Red Hat trademarked as commercial use and then also adding in Fedora Core.
Eventually named just Fedora, this system is sponsored by Red Hat but primarily oriented toward the Linux hobbyist. Despite initial reservations, it was quickly accepted and once again skyrocketed back to its status as one of the favorites.

This is commonly deemed one of the more innovative of the options available these days. Fedora has led the way in contributions to Linux as a whole and has made use of several enterprise-level features. However, it runs into some downsides as well—it requires some technical knowhow to

really make good use of it, meaning that it is still a better product for the hobbyists who enjoy tinkering rather than someone trying to learn.

Difficulty rating: Intermediate

Long-term support: Regularly updated

Pros:

- Innovative
- Secure
- Massive number of supported packages
- Strongly aligned with free software philosophy

Cons

- Enterprise rather than desktop features and usability
- Bleeding-edge features can sometimes be alienating to those who are less familiar with the system
- Requires a bit more finesse with Linux in general

Debian GNU/Linux

Announced in 1993, this was designed to become a non-commercial project completed through volunteer work of hundreds of developers. Most people thought it would fail—after all, how could hundreds of developers code for free in their spare time and create something cohesive? Nevertheless, the project was successful, and Debian became the largest Linux distribution within a decade.

It encompassed 1,000 volunteer developers' work, with nearly 50,000 binary packages for eight different processor architectures. From this effort came more than 120 other distributions based on Debian. No other Linux-based OS comes close to these numbers.
Debian exists in four main branches in several different developmental stages— "Experimental," "Unstable," also regularly known as sid,

“Testing,” and “Stable.” Thanks to this process, it is recognized to have quite intensive levels of quality control, and Debian has earned recognition as one of the best-established, tested, and bug-free distributions you can find.

Of course, the development process does have some down sides. In particular, the stable releases only come out every few years, and because they go through so much testing, they are not particularly competitive.
Difficulty rating: Good for beginners with some experience

Long-term support: Support is there, but new stable releases only come out every few years

Pros

- Quite stable
- Fantastic quality control
- More than 30,000 software packages
- Recognizes and supports more processor architectures than other distributions

Cons

- Newer stable software is usually not as cutting-edge due to the development
- Conservative due to the several processor architectures being supported,
- New releases are few and far between

Arch Linux

Designed for simple usability in 2002 by Judd Vinet, it was initially designed to be used for intermediate-to-advanced users of Linux who were already familiar with the OS. It did eventually gain popularity when promoted as a “rolling-release” distribution, meaning that it would only need the initial install to get it going.

From there, all that would be needed to keep the installation up to date was a monthly snapshot of install media to refresh the software. This was made possible by the package manager known as Pacman. It is also able to install software packages from source code, making it incredibly efficient, especially in tandem with its ability to simply pack up binary packages and the constantly increasing repository of tested software packages.

Despite its initial design for users with experience, thanks to the Arch Linux Handbook, along with all of the documentation of usage, it now allows for users with less experience with Linux to install, understand, and utilize it, and with the powerful tools it boasts, this distribution can be customized endlessly .

However, the rolling-release can cause problems, especially in the case of human error. Even a slight mishap can lead to a system that no longer boots after upgrading.
This means that, though there is plenty of information out there, this is still a system that will require some technical knowhow and experience to fix any problems that may arise.

Difficulty rating: Intermediate—need some experience to really use this well.

Long-term support: Regular rolling updates

Pros

- Fantastic software management structure
- Incredibly customizable with endless possibilities
- Plenty of online documentation and guidance available

Cons

- Can be difficult to catch on when a beginner
- Occasional breakdowns occur
- Breakdowns require experience to handle and fix

Slackware

Created in 1992, Slackware is the oldest Linux distribution that is still surviving. It was forked from a discontinued project and initially was released on 24 separate floppy disks and built upon Linux kernel verson 0.99pl11-alpha. It rose to popularity among the Linux distributions available, and by 1995, it is believed to have held a massive 80% of Linux installations.
However, as newer, an , and user-friendly options arrived on the market, the population of Slackware quickly declined. Despite the declination in popularity, it is still alive and kicking, and is enjoyed by those who are on the more technologically capable side.

Because it is not particularly customizable, it is quite clean and technical. It primarily utilizes a text-based system installer, and the package management system is quite outdated and primitive compared to more modern options available. However, because it is so simple and basic, it is also one of the cleanest of the Linux distributions you can get today, making it incredibly stable. Without enhancements, the chances of accidentally introducing a bug to the system are slim .

Between its reliability and lack of bugs, Slackware Linux is quickly becoming a core system—one that serves as the foundation for other software as opposed to complete distribution. Thought not competitive by any means at this point, it is still utilized by some today.

Difficulty rating: Advanced

Long-term support: Maintained, but not patched or customized

Pros

- Stable and mostly bug-free
- Adheres to Unix principles
- Largely unchanged from initial conceptualization

Cons

- Very conservative—no room for customizability

- Requires a lot of work to be used as modern desktop
- The upgrade procedure is difficult
- No official security support or updates

Gento o

Despite being the butt of several jokes floating around the internet, Gentoo is a valid option for people who know what they are doing. This particular distribution is quick and flexible while also entirely free. It is designed in particular for people with plenty of IT knowledge, such as developers and network professionals.

It is built with a source-based package management system known as Portage—a Python-based true ports system that comes with several advanced features of its own that can make Gentoo an attractive prospect to those who understand how to use it.

Originally created in 2000 by Daniel Robbins with a vision of creating a meta-distribution without needing any pre-compiled binary packages. He certainly succeeded—when you install Gentoo, you set up a system with the bare minimum to boot up, and from there, you must program it yourself. The system itself is incredibly customizable, and you are free to add and remove from it to make it work for you. Thanks to this, it is well-known for the flexibility it grants to those who choose to use it. They are able to create a wide range of experiences, and because of this, Gentoo can be found at the heart of several other projects, such as creating the foundation for Google's Chrome OS.

Difficulty rating: Advanced

Long-term support: Regular releases

Pros

- Flexible and customizable
- Can use several different compile-time configurations
- Can use several init systems

- Can run on several different architectures

Cons

- Requires a high level of knowledge to get started
- Takes longer to set up
- The process to upgrade packages through source can be lengthy

CentO S

Released in 2003, this is a community project that endeavored to rebuild source code for Red Hat Enterprise Linux into something that could be readily installed with regular security updates for everything included. Effectively, this makes CentOS a clone of Red Hat Enterprise Linux (RHEL), with the one difference between them being branding. Despite the fact that CentOS is a clone, there does not seem to be any real conflict between the two.

CentOS is deemed to be reliable as a server distribution—thanks to the utilization of the stable Linux kernel and software packages of RHEL. It has been deemed to be a reliable and free alternative to other server products that are available today, especially for those who are already quite experienced with Linux as a whole. These people find little difficulty in administrating the distribution.

CentOS is also usable for enterprise desktops as well—if you need something stable and reliable, with a preference for the support over cutting-edge software, this is the OS for you. Of course, if you do care about up-to-date and cutting-edge software, you may prefer to look elsewhere, as the updates to CentOS are a bit spaced out compared to some of the other options on the market.
Difficulty rating: Beginners with some experience

Long-term support: Regular updates are rolled out every 9-12 months, and new releases come out every 2-3 years

Pros

- Brings stability and reliability to the table
- Free software
- 7+ years of security updates
- Well-tested

Cons

- Does not utilize the most up-to-date technology
- Releases and updates do not always live up to their promises

Chapter 2 Quiz

Congratulations! You have made it through Chapter 2. Do you have an idea of which distribution you would like to try at this point? Try to answer these questions to ensure you understand the fundamental information before moving on to Chapter 3. The answer key will be on the page directly after this quiz.

1. **What is a Linux distribution?**
    - **a.** A book that tells you how to install Linux
    - **b.** A specific operating system programmed around the Linux kernel
    - **c.** The newest Linux hardware
    - **d.** All of the above
    - **e.** None of the above

2. **True or false: All Linux distributions are constantly changing**

3. **When choosing the distribution that is right for you, which features should you consider?**
    - **a.** User-friendliness and your own competency
    - **b.** How well it serves your own needs
    - **c.** The built in software
    - **d.** Whether you like the mascot
    - **e.** a, b, and c
    - **f.** a, b, and d

4. **True or false: Gentoo is great for beginners**

5. True or false: Ubuntu is great for beginners

Chapter 2 Answer Key

1. **B:** The distribution refers to the specific operating system program that you are using
2. **False:** Several distributions, such as Slackware, do not change much or have not changed in years.
3. **E:** While liking the mascot may be nice, it does not determine how well a program may meet your needs
4. **False:** Gentoo requires nearly constant programming to get it functioning at all.
5. **True:** Ubuntu is incredibly easy to get started with and is designed to be user-friendly.

Chapter 3: Install Linux

At this point, you should be ready to install Linux. However, you have yet another decision to make. Will you install your distribution on a virtual machine, or will you be running it on hardware? Even if you choose to install on hardware, will you run it as your primary OS, or will you create a partition for your chosen distribution?

If you are a beginner to looking into operating systems, this may seem intimidating, but try not to worry. This chapter will guide you through the basics of installing Linux on your device. When you are able to load up several different operating systems on one machine, you are able to toggle between them. This means that you can take advantage of everything your computer potentially has to offer through choosing exactly the software that will do whatever it is that you need to be done.

What you will get from this chapter, however, will be a solid foundation on what a virtual machine is and why you may choose to use them. You will be guided through the installation of Linux on your physical computer through the use of a USB drive, step-by-step. Then, you will be guided through the creation of a virtual machine on your machine running Windows, allowing you to effectively run an instance of Linux within the virtual machine itself. Lastly, you will be guided through the installation of Linux on a virtual machine on a computer that is currently running macOS. By the end of this chapter, you should feel comfortable with the idea of installing Linux onto your own device, whether you choose to install it on the physical machine itself or whether you prefer instead to run a virtual machine within the OS of your choice .

What is a Virtual Machine?

A virtual machine is an app on your computer that allows you to run an instance of an operating system within it. This allows you to effectively be running your particular machine’s native OS while simultaneously using another OS entirely within a window on your desktop. Virtual machines are not Linux-specific—you can run a macOS virtual machine on a Windows or Linux device, run Windows on a macOS or Linux device, or run Linus on a Windows or macOS device.

Effectively, when using a virtual machine, the app you are running functions as its own separate computer apart from the device that it is running on. The application creates the ability to run even virtual hardware devices. This means that your computer thinks that it is running on its own individual computer, monitoring its own processes, and using virtual hardware.

Unlike trying to run two operating systems on your computer, which requires partitioning, the virtual machine instead crates a virtual hard drive. This is a file that will be stored on your actual hard drive—this large file, typically multi-gigabytes, gets presented to the virtual machine as a hard drive, allowing you to bypass needing to create any sort of partition or anything else that may become complicated with the hard drive. Do keep in mind that, while a virtual machine is great for creating virtual instances of an OS for your usage, it is notably slower than running a proper OS. Because of the extra steps that go into supporting the virtual machine, it does have some further processing power required. This means that a virtual machine is most likely not your best way to be playing games of any sort.

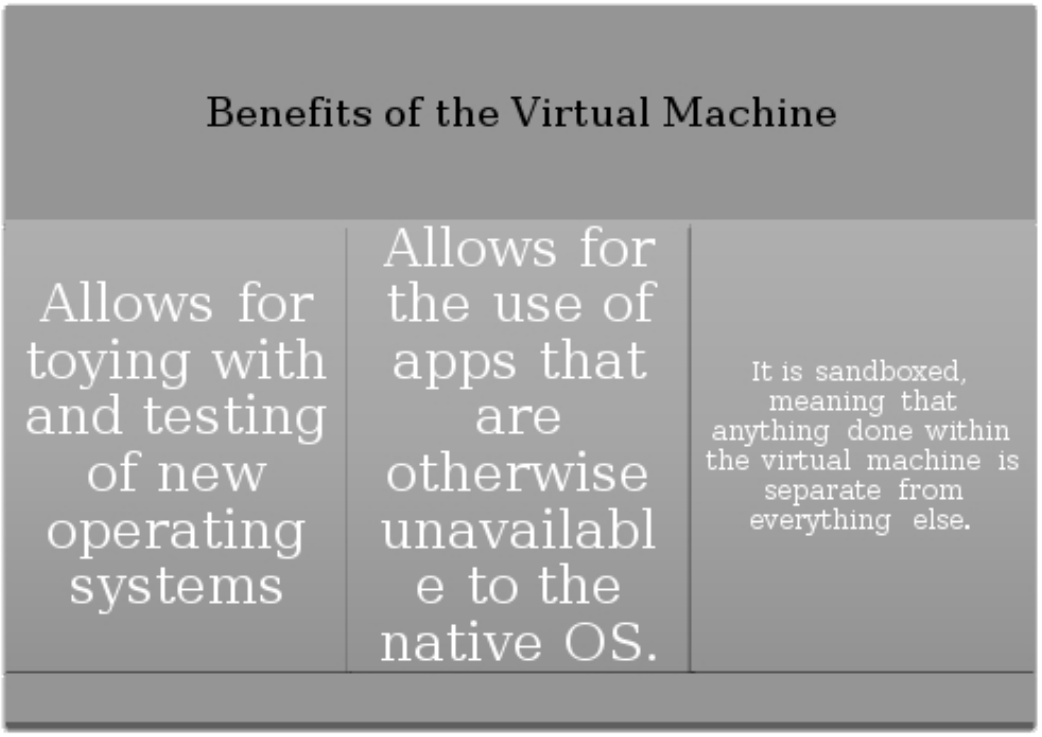
As a major pro, however, the only limit to how many virtual machines you can have installed is only limited by the amount of physical hard drive space you have available to you. If you have enough room, you can continue to add more installations of several other operating systems. However, you must keep in mind that each of these installations will take up at least several gigabytes of space within your hard drive. You can also choose to run several virtual machines at the same time, though again, you will find that the upper limit is your physical hardware. Your virtual machines will each need access to some of your processor, RAM, and other resources, meaning that the more you are actively trying to use at any given time, the weaker the running will most likely be.

There are several reasons that someone may choose to run a virtual machine. For some, this could be done just to mess with. If you are particularly interested in computers and how they work, you may choose to use a virtual machine to play around in other operating systems that you are not actively running as your primary OS. This means that you can mess around with a new OS before deciding if you really want to install it

physically, allowing you to, for example, play with several different Linux distributions before you chose which you prefer to use.

Other people may find that they have very specific needs from time to time that requires a limited app that is only available with a specific OS. For example, if you are running Linux, but need an application that was primarily contained within Windows XP, you could choose to run Windows XP on a virtual machine to make use of that specific app .

One other common reason that people use virtual machines is the fact that the virtual machine is sandboxed. This means that it is all contained within the virtual machine itself, so there is nothing that you can do within the virtual machine that puts the rest of your system at risk. Like a sandbox and its sides to contain it, the virtual machine is entirely sequestered within the hard drive and will not be impacted by whatever it is that you have chosen to do within it. This means that you can test apps that are being developed in a safe space without worrying about the repercussions that would occur if the entire system failed.



Installing Linux on Physical Hardware

Before it is time to worry about the virtual machine, however, you will first learn how to install Linux on physical hardware. When using this process, you are creating the option to install Linux on your computer as a separate partition.
This means that you will have two (or more) operating systems on your machine. This process will be given to you step by step.

Step 1: Downloading the Linux distribution
Did you choose out distribution in the previous chapter? If so, this is what you will be searching. If you have not yet decided which you prefer to use, you will need to spend the time making your decision. If you are reading this book, you are most likely new to Linux as a whole, and because of this, you will find that you are best served by a lightweight distribution that will be a bit more user-friendly, such as Ubuntu or Linux Mint. Of course, you can also choose to challenge yourself and go for something a bit more intensive if you so choose. Before beginning, also make sure that you have a CD or a USB drive that you can use for the installation. Because CD drives are largely becoming obsolete in many consumer-grade products, the USB is most likely going to be the most readily-accessible form for any computer.

Once you have chosen the distribution that you will use, you must download it. You can find the distro free to download in an ISO format. This is an archive file that has an identical copy of data that was found on a disc and is the most common format to distribute Linux. The identical copy is known as the image and is what you are seeking to copy in order to install the software.

1. Download the ISO for the distribution of Linux you have chosen on the distribution's website.
2. Prepare the USB to become bootable with a program such as Rufus.
3. Install an image burning program such as Pen Drive Linux or UNetBootin.
4. Load image onto the formatted USB.

Step 2: Booting into the live USB

Now, you are tasked with loading into the live USB in the first place. Most computers will set to boot, starting with the hard drive, so you will need to make some changes to the default booting order before it makes the necessary changes. This process is not as intimidating as it sounds—all you need to do is reboot the computer, enter the boot menu, and choose the source that you would like your computer to boot from.

Sometimes, the boot menu is hidden from view, but it is still accessible. In these situations, you will need to access the BIOS menu in order to gain access to the boot menu, which can be done at the splash screen. The command necessary to load into the BIOS menu should be displayed in one of the bottom corners on the splash screen.

1. Reboot the computer
2. As the computer reboots, press the key to trigger the boot menu. You should see the necessary key in the corner of the splash screen—the screen where the manufacturer's logo is displayed during the initial startup. Common keys to access the boot menu are F12 or Del.
    a. If you are using Windows 8, try holding the Shift key while clicking restart. This triggers the computer to boot with the Advanced Startup Options, allowing you to make the decision.
3. When in the boot menu, select the live USB or CD that you are using to load up Linux. Upon choosing your method, save and exit the boot menu or BIOS. Your machine should continue with the booting process with your selected settings.

Step 3: Try Linux before installing (optional)

Trying the version of Linux that you are in the process of installing is optional, but strongly recommended if you are not yet familiar with Linux or you are unsure which version to install. Your live USB should allow you to launch what is referred to as a live environment, allowing you to test it prior to making any sort of changes. While you will be unable to create files during this stage, you will be free to navigate the system and experience the

interface to make an informed decision before spending the time to install and format.

Step 4: Starting installation

The installation process itself is quite simple. From the boot menu, you will have the option to install rather than launch. At this point, your machine will handle the bulk of the work, but you will be prompted to configure basic options, such as the language that you will be using with the OS, the keyboard layout, and the time zone you are in.

Step 5: Creating a use r

At this point, you will be prompted to create the login information for your OS. You will insert your name, your computer's name, the username that you are choosing for this instance, and a password for logging in and confirming that you are authorized to perform any administrative tasks.

Step 6: Creating the partition (optional)

If Linux is your primary OS, you can skip this step. However, if you are running multiple operating systems on your machine, you will need to separate out a partition specifically for your version of Linux. The partition is the portion that has been formatted in the way that the specific OS requires, allowing for the OS to run as intended.

Some more beginner-friendly distributions, such as Ubuntu, will format the partition on their own, though you can adjust the size manually. Keep in mind that most installations will need at least 20 GB, so you must make sure that whatever partition you set will accommodate the OS *and* any other files and programs that will be stored within it.

Sometimes, the installation process will not automatically configure the automatic partition, and when this happens, you must make sure that the partition is in the format Ext4.

Step 7: Booting Linux

When the installation finishes, it will automatically prompt you to reboot. Upon booting, you will see a different screen. This will pull up the bootloader, which will prompt you to pick the distribution you are installing. One of the more likely bootloaders that you will encounter is GNU GRUB. Of course, if you have opted to skip the partition stage and load your Linux distro as your only OS, you will likely not see this screen automatically. As with before, you can trigger the bootloader by pressing the Shift key immediately after the manufacturer splash screen loads.

Step 8: Checking hardwar e

While most hardware should be fine immediately after installation, you may find that some of the hardware you are using will require you to download some drivers. In particular, graphics cards are notorious for requiring extra drivers. You can usually find some sort of open source driver to get the job done, but if you want to really optimize your GPU, you will need to get the proper drivers from the manufacturer.

Step 9: Using Linux

At this point, your installation should be complete, and your hardware should be functioning. At this point, you are ready to begin using your new OS! You may choose to go download programs that will allow you to optimize your usage of the program, or you can go on your way using it.

Installing Linux on Virtual Machines on Windows 1 0

Sometimes, the method you will be using to install Linux will utilize a virtual machine. In particular, you may choose to use a virtual machine to sandbox software in order to ensure that you do not damage any of the files on your primary OS. You may also wish to mess around in the operating system a bit more before taking the plunge and installing it. Luckily, using a virtual machine on Windows does not need to be difficult. 64-bit versions of Windows 10 Pro, Enterprise, or Education already are preloaded with a proprietary virtual machine known as Hyper-V, meaning that you will not need to go out of your way to get more software. If you do not have Hyper-

V already, you will need to find a VM program. A free option that you can use is Oracle VM VirtualBox.

Step 1: Installing the virtual machine
Start by verifying that you already have a virtual machine installed. Do this by searching for “Hyper-V” on your computer under Windows features. You should see boxes on the left-hand side. When you check those boxes next to the files, you are turning files on and off. Click to enable Hyper-V if you can find it on your computer. Upon installation, you will need to reboot.

If you do not have Hyper-V, search for another VM program that you seem to like online and install it.

Step 2: Setting up the virtual machine
At this stage, you must make sure that you have enough hard drive space available for the OS and files of your choice, and you will need the ISO file you intend to use. As in the previous version, the ISO can be downloaded from the distributor.
If you are using Hyper-V, searching for Hyper-V Quick Create will give you a sort of shortcut to ensuring that everything is set up as necessary. All you will need to do is follow the prompts on the screen within the program, selecting your ISO file if you do not wish to use one of the provided ones.

Without Hyper-V, you will find the process to be slightly less streamlined, but still quite manageable overall. Most virtualization software will be quite similar—it will be a matter of preference and availability to determine which you prefer.

1. Upon launching VirtualBox, you will see an empty menu. From there, you will need to click on the NEW icon in the top left of the program’s window. You will then name the VM and then select which kind of OS you will be using.
2. Next, you need to allocate memory to the VM, providing it with the necessary resources. Typically, the best bet is to going to be to take the recommended amount and then click next.

3. The next prompt you will see will ask if you want to add a virtual drive. Agree to this, click Create, and select VDI on the list. VDI stands for the virtual disk image. Choose the Fixed Size option on the next page. At the end, you will be asked to name the drive you are using and then confirm the size.
4. Click create, and the virtual drive has been created.

Step 3: Installing the OS

At this point, it is time to install your chosen distribution of Linux. Within the VirtualBox window, select the Start arrow. It should open up a window that will be titled Select start-up disk. Click the folder on the right, next to the drop-down menu, and here, you can select the ISO file for your distribution. From there, all you do is click Start, and the OS should be installed.

Step 4: Shutting down the virtual machine

When you get to the point where you are done with the machine, resist the urge to just click out of the window. Instead, click on the File menu and select the option to close while selecting 'Save the machine state.' This ensures that you will keep the information within the virtual machine .

Installing Linux on Virtual Machines on macOS High Sierra

At the time of writing this book, the current macOS is High Sierra. Installing Linux through a virtual machine on High Sierra is largely similar to how you would go through the process on Windows. Considering, at this point, you have been guided through the most difficult processes, this will be the shortest guide of them all. All you will be doing that is new within this chapter is installing VirtualBox on the macOS. From there, you can follow the steps detailed in the previous subchapter.

Step 1: Run VirtualBox

1. Start by running the VirtualBox installer. You may see a warning that that macOS does not allow unsecure system extensions. Just click ‘Next’ and keep going.

2. At the end of the installation, it will fail. Don’t worry about the failure; just keep following this tutorial.

3. Go to System Preferences > Security & Privacy. At the bottom, there should be a prompt that says that the system software was blocked. At the bottom left corner of the window, click on the lock. It will ask you to put in your password. Do so and then click ‘Allow’ next to the message telling you that the program was blocked from loading.

4. Reload the VirtualBox installer. It should go through this time.

At this point, the process is largely the same as opening a VM in Windows. The only differing point between the two was the installation process.

Chapter 3 Quiz

Congratulations! You have made it through Chapter 3. At this point, you should be prepared to load up Linux on your own devices. Try to answer these questions to ensure you understand the basics before moving on to Chapter 4. The answer key will be on the page directly after this quiz.

1. **Which tools are necessary to install Linux (either on hardware or on a virtual machine)?**
    - **a.** ISO file
    - **b.** USB drive
    - **c.** Virtualbox
    - **d.** A and C
    - **e.** All of the above
    - **f.** None of the above

2. **What is necessary to run several operating systems on the same system?**
    - **a.** Virtualbox
    - **b.** A USB drive
    - **c.** A partition in the hard drive
    - **d.** A computer with two hard drives
    - **e.** All of the above
    - **f.** None of the above

3. **What is a virtual machine?**
    - **a.** A computer capable of running operating systems within operating systems at the same time
    - **b.** A way to keep an OS partitioned away from the other parts of the computer's OS
    - **c.** A method that can be used to test software
    - **d.** All of the above
    - **e.** None of the above

4. **True or false: You need a virtual machine if you want to run Linux**

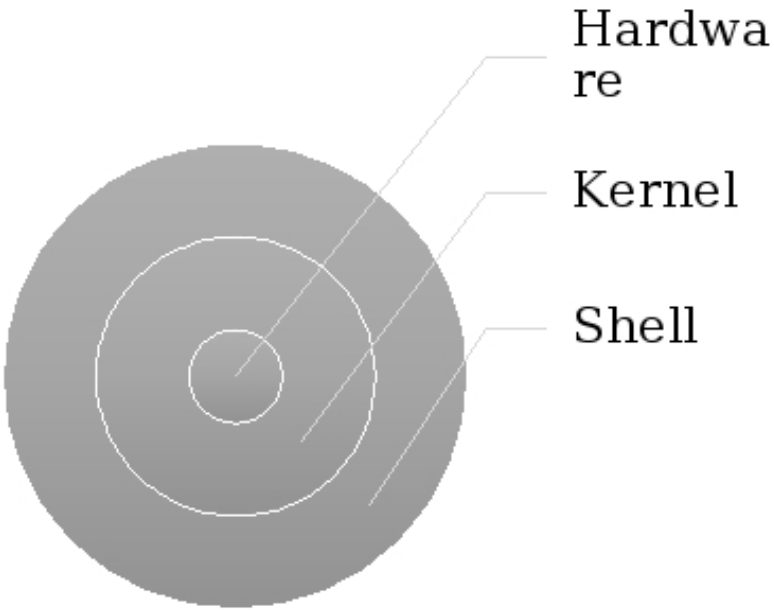
5. **True or false: the USB drive that you will load the ISO onto will be ready out of the box**

Chapter 3 Answer Key

1. **A:** The only listed component that is necessary to run Linux is the ISO file that is used to distribute the OS. The USB drive can be bypassed when installing on a virtual machine, and Virtualbox can be bypassed when installing on hardware with a USB.
2. **C:** The only necessary component listed is the partitions between operating systems. Virtualbox is one specific virtual machine, but there are several that will work in its place, the USB drive is not necessary, and you do not have to have two hard drives—the whole point of the virtual machine is to partition a single hard drive to accommodate several operating systems.
3. **D**
4. **False:** You can install Linux onto the hardware, bypassing the need for a virtual machine.
5. **False:** You must format the USB drive to be ready to accommodate the ISO.

Chapter 4: Linux Shell

With Linux freshly installed, it is time to begin interacting with your new OS. This is primarily done through what is known as a shell. The shell allows you to interact with the operating system, which at this point, you should understand is crucial for the functioning of your computer. Your operating system consists of several layers: at one end, you have the kernel, which is how the OS interacts with the hardware of the computer, and on the other end of the process, you have the shell, which the user interacts with.



Stop and consider this graphic for a moment: Note how the hardware is at heart. From there, you have the kernel that surrounds the hardware, and the shell that engulfs the kernel. This allows you to sort of see the hierarchy—you cannot access the innermost levels without first going through the external levels.

Within this chapter, you will learn what the shell is as a system, as well as what its primary functions are. You will learn how to access the shell through the terminal. You will be guided through shell scripting, as well as be provided with basic command line editing that you need to know in order to interact with the shell. Lastly, you will be walked through the types of shells available.

What is Shell?

Computers primarily operate on a binary—this is the usage of 1s and 0s to represent whether a switch is on or off. However, if you were asked to put your input into your system via binary, you would find yourself spending far longer than necessary trying to translate and input your code. This is where the shell comes in.

Thanks to the shell, you do not have to master binary. The shell acts as the translator between human-friendly commands written in English and then translates them into something usable by the kernel. In translating everything for the kernel, you are free to memorize commands that are far simpler and more manageable. This also allows you to input your commands via a keyboard. Overall, the shell has several key features, such as:

- **It is a user program:** It is an environment that is designed to accommodate human user interactions
- **It is a command language interpreter (CLI):** It receives and executes a command from standard input, such as a keyboard, without the need for the memorization and translation of difficult program languages such as binary.
- **It is automatic:** As soon as you open a terminal, the shell will begin functioning
- **It works with the kernel:** Though it is not a part of the kernel, it does work in tandem with the kernel in order to allow for the execution of programs, creation of files, or completion of any necessary commands.
- **It is versatile:** There are several different shells available for Linux, which will complete the same job, though they use different syntax and have varying built-in functions.

Gaining Access to the Shell

When you are interacting with a shell, there are several different methods through which you can use. The most common accesses to the Shell include the terminal and a secure shell connection. These will come with varying degrees of permissions and abilities. Most commonly, you will find that you are interacting with Bash, though sometimes, you may encounter other forms. Nevertheless, understanding how to access the shell is necessary.

Installing Bash on Windows

This section will require you to install Bash through the Git for Windows installer. It is providing straightforward instructions to you that may not seem particularly informative until you have the screen in front of you. Nevertheless, return to this section when you are ready to install Bash if you want the quickest way to do so.

1. Terminal access to shell requires you to first install Bash. This is quite simple—download the Git for Windows installer. When running the installer, click on the run, and then on the next five screens, you will click "Next."
2. When presented with the option, select "Git from the command line and also from 3rd party software" and then click "Next." This selection will allow you to access via command line, which is what you need. If you mess this step up or miss the selection you need, restart the installer and change the options accordingly.
3. Click "Next again."
4. Select "Checkout Windows-style, commit Unix-style line endings" and then click "Next" again.
5. On the next screen, select the second option. You should select "Use Windows' default console window" and click "Next."
6. Click "Next" once more.
7. Install.
8. When the install has completed, select "Finish."

This should leave you with Git and Bash available. Bash is the default shell used in macOS X and in most forms of Linux, so you will not need to go

through this process if you have one of those operating systems already installed on your computer .

What is the terminal?

A terminal is the CLI used to access Bash. As with the shell itself, Linux and Mac users will already have a terminal program installed.
You have also installed your own terminal access in the previous section if you used Git for Windows installer—Git is the commonly used terminal for Windows users.

The terminal will be the way that you are accessing your computer. When you use Bash within the terminal, you are able to perform far more tasks than you have ever had to do before on your computer. This will also allow for the performance of several other tasks in an efficient manner, such as monitoring the current working directory, changing directories, creating new directories, and more.

Terminal access to shell

When you are ready to access Bash, you must use the right program for the program you are using. In OS X, it is usually referred to as Terminal. In Windows, it is Git Bash. In Linux, it is typically just referred to as Bash .

When trying to access your shell within Linux, the easiest way to do so is to run a terminal emulator. There are several terminal emulators available to you—these are programs that will emulate the terminal hardware that is typically used in Unix. They are then mimicked and utilized within Linux, being presented as a window for you to access and interact with. Upon accessing the emulator, you should have options to configure the terminal, allowing you to change the text and colors or other personal settings.

Most of the time, GNOME is available, especially if you are running Bash, though others also exist. Some other commonly used terminal emulators include xterm, Konsole, and LXDE.
GNOME can be found within the applications menu. Running it should bring up the terminal for you, which you can then begin to interact with. When you interact with the terminal, you will enter your commands, and hitting the Enter button on your keyboard should trigger it to send .

When you are in the terminal, there are several commands that will be critical for you to know, which will be discussed in Chapter 5. Before beginning, please recognize that your terminal will display the dollar sign ($) when it is waiting for input. Other shells may use a different character, but if you are working within Bash, it will be the $. This will let you know that you should insert your own commands.

Secure shell connection

Sometimes, there is a need to connect to shell accounts on a remote server. So long as the server that you wish to access is running the proper SSHD server software that will be able to manage and accept any connections, you should be able to use what is referred to as an SSH program. These grant a remote connection.

When you have established a secure SH connection, the shell session will begin, and you will have the ability to control the server via typed commands on your local computer. This has several uses, but most commonly, it is utilized by network and system administrators. If you have a need for a secure method to remotely manage a computer, you will find SSH to be quite beneficial to you .

In order to create the SSH connection, there are two necessary components: The client and the server-side component. The client is the application that you will use on your computer.
You will use the client to connect to the other computer or server that you wish to control. The client then uses the information provided to initiate a connection. Through the verification of credentials, an encrypted connection is created.

The server must have a component known as an SSH daemon. As a brief reminder, daemons are background processes that are activated and then remain to wait for any commands that may be relevant to them. In this case, the SSH daemon is always on the lookout for a very specific TCP/IP port that informs it of a client connection request. The client is responsible for starting the connection, and then the daemon will use its own software to directly interact to verify credentials and then allow for the SSH to be established.

In order to establish this connection, you will need to ensure that you have the corresponding client and server components installed. A commonly used open source SSH tool is OpenSSH—this will require you to grant access to the terminal on the server and the computer that you are using to allow for the connection to be maintained. If you have chosen Ubuntu, this will not be installed by default, but you may already have it on other distributions of Linux.

You can check quite simply whether you have an SSH client already installed through the following process:

1. Load up an SSH terminal. You can do this by searching for "terminal" or pressing **CTRL + ALT + T** on your keyboard.
2. With the terminal brought up and within it, type in **SSH** and press **ENTER.**
3. If the client is already installed, you will see your access to the other server. If you do not get a response with options and commands listed for you, you will need to install OpenSSH.

Luckily, installing OpenSSH is also relatively simple. You can do so with just a single command:

sudo apt-get install openssh-clien t

Upon typing in this command, you should be asked for a superuser password. Provide this and press Enter when prompted to complete the installation. You should now have SSH access to any machine that has the server-side application, so long as you have the credentials to prove that you should have those privileges.

Of course, you still need to install the server side component as well in order to truly get access. Again, ensure that there is not already the SSH component installed on the server.

1. From the server machine, open the terminal. Again, you can either search “terminal” or use the **CTRL + ALT + T** command from the keyboard.
2. With the terminal up, type in **ssh localhost** , followed by pressing Enter.
3. Systems without SSH will respond that the connection has been refused.

When the system is refused, you know that you need to install OpenSSH. With the terminal still open, you will want to run the following command:

```
sudo apt-get install openssh-server ii .
```

You should again be prompted to enter the superuser password. Then, hit **Enter + Y** to grant the installation permission to continue right after the disk space prompt. It should then automatically install all required files. In order to test that you have been successful, try the following command:

```
sudo service ssh status
```

If you have done your job, you should receive a response within the terminal that informs you that it is active and how long it has been running. Congratulations! You have stablished your SSH connection !

Types of Shell

When you are using Linux, you will primarily encounter two types of shells: These are the Bourne shells and the C shells. Each category of shells will involve slightly different command standards, but they will still get the job done of allowing you contact with the shell to interact with and control the computer.
By and large, you will primarily be utilizing the Bourne Again SHell, known as BASH. However, it is never a bad idea to familiarize yourself with the several different options out there for you to utilize for future reference.

Bourne shell (sh)
The Bourne shell is the original shell that had been used with Unix. The prompt that you will see when using the Bourne shell is the $ symbol—this tells you that the terminal is ready for input. There are several other shell types that are recognized as related to the Bourne shell, such as the Korn shell and bash.

- **Korn shell (ksh):** This shell was invented by David Korn in the mid-1980s. It is nearly completely compatible with the Bourne shell, meaning that if you are already using the Bourne shell, you can utilize this particular version immediately. Likewise, a device utilizing the Bourne shell can utilize the Korn shell instead, allowing for versatility. This is a popular shell, thanks to its compatibility with the Bourne shell and the fact that it also offers several desirable features from the C shell while including its own benefits as well. It includes command-line editing, allowing people to change mistakes simply, something the Bourne shell does not offer at all. Along with this, ksh utilizes job control—the ability to stop, start, and suspend commands at the same time.
- **Bourne Again shell (bash):** This is the free distribution of the Bourne shell that comes with Linux systems. It shares a lot of similarities to the Bourne shell that it came from, though some people tend to struggle with it. Luckily, there are several alternative guides that you can use to work through the process of using this.

C shell (csh)
Similarly to how the Bourne shell is deemed the defining shell of a family of others, C shell is deemed to be its own category of shells. It draws its name from the fact that it is programmed in C, and it runs in a text window in which you can insert your commands, which are then read by the program. It boasted features such as interactivity and style, with these making it easier to use, and the language was deemed to be more readily understood by those who were trying to use it. In particular, C shell had several important features, such as its grammar and syntax, allowing for it to be readily followed, and it allowed for edits to be performed. Unlike the

Bourne shell, the C shell will utilize the percent (%) symbol as the prompt for input.

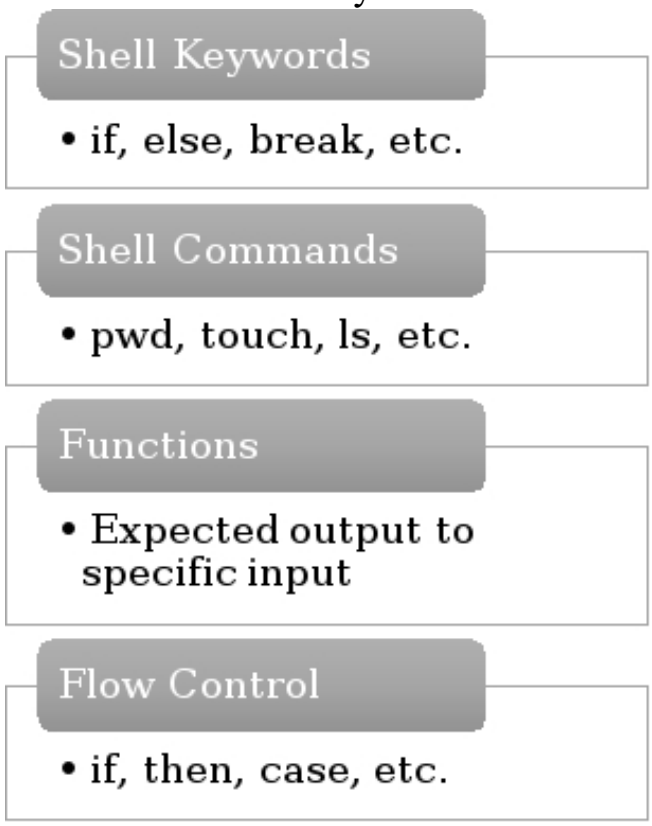
- **TENEX/TOPS C shell (tcsh):** A derivative of csh, tcsh is entirely compatible with chs. It is more-or-less the csh, though it added in some improvements and enhancements that allowed for more uses for the program. In particular, it allows for command line editing, filename/command completing, and more. Thanks to these features, it is great for those who struggle with traditional Unix commands, or even for those who are slow when it comes to typing. Since the filename will try to complete itself with the press of the Tab key, you are able to type in half of the file name, hit Tab, and have the program try to fill it in for you.

Shell Scripting

Shell scripting, then, is the composition of several commands for the shell to execute. These can combine several repetitive tasks and create one simple script that can remain in storage until it is necessary to utilize, allowing for execution at nearly any point in time. This allows for the reduction of effort required by the end user since the entire series of repetitive tasks can be combined into something shorter. This is perfect if you will be using the same long series of commands routinely. If you combine them all into one script, you can trigger the whole thing to run at the same time, rather than having to write out each command every time you use it.

Remember that shell scripts require a very specific syntax. The shell is comprised of several elements that come together: the keywords, commands, functions, and flow control. Each of these elements plays their own roles and are crucial to understanding in order to truly create the language in a way that the computer can process it. Think about your own language for a moment—word order matters, right? “I can have pizza” and

“can I have pizza” have two entirely different meanings—one is confirming permission while the other is asking for permission. Similarly, if you mix up your syntax, your shell will not always be able to understand you.



How to shell script

For the purpose of this tutorial, we will be working with bash standards, as that is the standard structure that is commonly used within Linux systems by default.

The steps to creating a Shell Script are not particularly difficult, so long as you are able to follow along. There are five simple steps that you will need to follow in order to create your script. Of course, your scripting is also going to grow more complex as you continue through this book and learn more commands to create more sophisticated scripts in the first place. Consider this a basic guide.

First, you must create your file using a vi editor. You want to make sure that the file is named with **extension .sh** to make sure that it will function.

Now, you must begin the script, utilizing the following code:

```
#!/bin/s h
```

The symbol **#!** Is an operator. It is referred to as a shebang, and it directs the script to ensure that it gets to the right shell. When you then direct it to /sh, you are directing it to the Bourne shell.

After you have directed to /sh, you are free to write the code that you want to have executed. This is where Chapter 5 will come into play—you will learn the commands necessary to create exactly the script that you are looking for or need. When you do finish writing your code, safe the script file in .sh format. Let's say your file is named listfile.sh because you have designed it to list out all of the files present at that point. When you want to execute your saved script, you will type:

```
bash listfile.sh
```

Typing that command will then trigger the script to execute, providing you with the end result. For now, this is all you need to know. As you read through Chapter 5, start to think of some of the potential options for coding the script that you will want .

Basic Command Line Editing

As one final note before wrapping up the chapter on shells, you will be provided a short list for command line editing. This feature can be incredibly helpful to you if you are able to memorize theses commands.

CTRL + L: Clears out the screen

CTRL + W: Deletes the word the cursor is currently on

CTRL + U: Clears out the current command line

Up or Down arrow keys: Recall commands

Tab: Auto-completes file names and directories

CTRL + R: Allows the user to search through the command history

CTRL + C: Cancels the current commands

CTRL + T: Reverses the last two characters before the cursor

ESC + T: Reverses the order of the two words preceding the curser

CTRL + H: Deletes the letter in front of the cursor

Chapter 4 Quiz

Congratulations! You have made it through Chapter 4. At this point, you should have a pretty solid foundation for understanding the Linux shell. Try to answer these questions to ensure you understand the basics before moving on. The answer key will be on the page directly after this quiz.

1. What is a shell?

a. The kernel

b. The physical case around the hardware

c. The program that allows user interaction with the OS

d. The operating system's type

e. All of the above

f. None of the above

2. True or false: Bash is necessary for Linux to run.

3. What does the terminal allow you to do?

a. Interact with the operating system

b. Control the computer's functions

c. Establish ssh connections

d. All of the above

e. None of the above

4. Which two shells are compatible with each other?

a. Bourne shell and C shell

b. Bourne shell and tcsh

c. Bourne shell and Korn shell

d. All of the above

e. None of the above

5. True or false: The shell is the innermost part of the computer

Chapter 4 Answer Key

1. **C:** The shell is the program that the user will utilize to interact with the system.
2. **False:** It is useful if you wish to have any real control over your computer system, but it is not absolutely necessary. Even if you needed a shell, bash is not the only option out there for you.
3. **D**
4. **C:** Both Bourne and Korn shell users can switch between the two interchangeably.
5. **False:** The shell is the outermost component of the operating system.

Chapter 5: Linux Commands

This chapter is designed to be a sort of accessible dictionary for you, where you can access a wide range of Linux commands on demand in one place. Because, especially at first, you are going to find that you are not quite familiar with the system commands off the top of your head, it can be useful to have a chapter to fall back on to look up the command you will need.

These commands are designed to be used within the terminal, allowing you to directly communicate to the system itself. This means that, through the use of the terminal and the commands provided within this chapter, you will be able to directly influence and interact with the computer. Of course, there are also several rules that you will have to remember when utilizing these commands.

Keep in mind first and foremost that $ is prompt to inform you that the individual is officially ready for your input within bash. However, csh will utilize the % to prompt you to enter your input. Also, remember that the inclusion of a colon after the code within this book is done to separate it from the description of that particular command's function, and the colons should not be included.

Beyond that, you must also recognize that Linux is, in fact, case-sensitive. If you enter a code with a capitalized letter that is not intended to be there, it will fail to load up whatever it is that you have prompted it to load in the first place.

This does come with some advantages, such as allowing for identical strings of symbols in the same case to be processed quicker rather than having to process all of the letters to ensure that the letter itself is registered in either case rather than the specific case expected. Even though computers today are more than capable of processing that extra load, a habit has won out, and case-sensitivity remains to be a feature to remember .

System Information Commands

Hardware is incredibly important to be familiar with when you are working with the more intricate matters of a computer. For this reason, Linux comes with several commands that will allow you to access any information about the hardware that you may need. These commands will help you understand the configuration details of several hardware components.

When you use these commands, you will be able to track your current system's ability to function, and this means that you can see how to allocate resources or know how well the system can handle your current tasks. While some distributions will have slightly different commands, these are the most readily available commands out there if you need to gather information about your system.

As you read through these command codes, notice how there are several that seem quite similar, but they all have their own nuances that set them apart somehow.
It can be intimidating to see all of these codes typed out in front of you, but remember, there are tools available to you that will help you if you cannot or do not wish to memorize the commands .

accept: Allows you to accept a job to a destination

arch: Displays the print machine's hardware name

date: Displays the system's date and time

df: This particular command triggers the computer to report the disk space available in the file systems. It reports on several different partitions and their mount points, providing data for all of it.

fdisk: This is the utility command to change partitions on hard drives if necessary to do so.

free: This tells your computer to check the amount of RAM on your computer—free, used, and the total amount of RAM on the system.

hdparm: This tells your computer to provide you with information about hard disk drives or other sata devices.

hwclock: Displays or allows for the configuration of the hardware's cloc k

hwinfo: This is a general purpose command that probes the hardware—it will provide you with either detailed or brief information about hardware components, providing more than what you may use lshw for.

lnxi: This is a bash script, meaning it contains a larger code. In fact, this particular command has a 10,000 line code within it. In activating this code, you will gather up all sorts of hardware data from several different sources, placing it all into a nice, easy to read a report that even beginners will have little issue navigating.

lsblk: This triggers the device to list out block devices. In doing so, you will see hard drive partitions as well as optical and flash drives.

lscpu: This command will pull up a report about the CPU and processing unit on your computer.

lspci: This command lists out pci buses while also giving you further details about them. It includes the vga adapter, graphics card, USB ports, and more. With this particular command, you can also filter out the specific information you are looking for thorugh the utilization of **grep** , which will be discussed in further detail later in the chapter.

lsscsi: This command lists out scsi devices, such as hard drives and optical drives.

lshw: This is a general purpose command that will list out all of the hardware installed on your system, as well as the data included within it.

lsusb: This command provides the user with data about the USB controllers and any devices connected to them. You can add in **-v** to your code if you wish for this detailed information to be printed for you.

mount: This is designed to either mount or unmount devices, as well as view mounted file systems currently interacting with the server. Once again, you can utilize **grep** to filter out exactly which files you are interested in seeing .

uname –m: triggers the system to print the machine hardware name

uname -r: triggers the system to print the kernel release being use d

System Shutdown, Restart, and Logout Commands

Thankfully, Linux makes it incredibly user-friendly to restart your computer or shut it down. While on most other operating systems, you simply go through the buttons and prompts to select shutdown, restart, or logout, with Linux, you can do this via the terminal as well. For those who are always accessing their terminal, this means that you will be able to manage your system with just a few keystrokes.

Keep in mind that most of these processes will end your current session—for that reason, you must make sure that you are ensuring that you will not lose your work as you go. This section will guide you through the most common shutdown commands. With these commands, you can begin to take control of the system—you can program the system to shut off at a predetermined time or decide to turn off within a few hours or minutes. No matter when you choose as your computer's shutoff time, you will be able to program it in. This is the beauty of Linux—you can program whatever commands you desire within it .

sudo shutdown –h HH:MM: After the –h, you can enter a time that you would like to trigger the computer to turn off. You can enter **now** or in any other time, keeping the format in **HH:MM** , so if you want your computer to turn off at exactly 7:38 PM, you will enter:

sudo shutdown –h: 19:38

Without the time included, you will get a message telling you that you must set a specific time for the shutdown to occur. Instead of using HH:MM or **now,** you can also opt to do **+30** to add 30 minutes to the program.

sudo shutdown –c: this will cancel the currently scheduled shutdown command if you have it set to cancel at a specific point in time. When you

use this, you will be able to ensure that you are not interrupted by the scheduled shutdown.

sudo reboot: this will trigger your computer to restart itself .

init Service manager. This will determine the level at which your program is running. There are several different run levels that can be within—from 0-6, s, and m.

- **init 0:** This is shutdown
- **init 1:** This is the single-user mode or emergency mode .There is no network or multitasking.
- **init 2:** There is no network at this stage, but the support for multitasking is available.
- **init 3:** The network and multitasking are present, but there is no GUI.
- **init 4:** Similar to init 3, but used in some research. This state is essentially irrelevant to you right now.
- **init 5:** The network is multitasking, and GUI is present.
- **init 6:** This is used to tell the system to restart.
- **init s:** This tells the system to enter maintenance mode.
- **init m:** This tells the system to enter maintenance mode. Synonymous with init s.

pkill: This will kill a process by name. Killing the processes before shutting off the system is necessary with Linux, or you risk causing issues with the software and files. You can use this to end the sessions of other users as well. When you use this to kill other users, make sure that you do not kill a root user or a system level process or you will kill the server's current processes. To kill the process of another user, use the following command:

```
pkill –KILL -u {username}
```

kill: This will terminate a process.

logout: This will force the logout of a shell. You can also use this if you are a regular user within the computer. It will log you out of the sh or ssh .

Files and Directory Commands

Sometimes, you need to find something within your files and directory. Luckily, rather than having to dig through everything line-by-line, there are several systems commands you can utilize in order to bring it all to the forefront of the terminal for you to access as needed. This list for you is provided in alphabetical order so you can find the commands by spelling as necessary.

cat: This will print the content of a fiile for you. For example, if kitten.txt has a long poem about why kittens are cute and fluffy if you use the following code:

```
$ cat kitten.txt
```

you will end up with the long poem about cute and fluffy kittens in your terminal.

cd: This command allows for the changing of the directory to access a new one. For example, if you use the following command:

```
$ cd /
```

You will be moved to the root directory. So long as you define the directory that you wish to reach, you should be transferred there.

cd . . This command allows you to move up one directory level. If you are currently looking within a directory named a file, and within that directory, there is another named basket, for example, using this command will move you from file to basket.

cp [file 1] [file 2]: This command allows for the copying of one file into another one. If the name of file 2 is nonexistent on the system, it will simply be copied over. However, if it already exists, the content sharing the same name will be overridden. For example:

```
$ cp kitten.txt puppy.txt
```

In this example, so long as there is no other file named puppy.txt, the designated kitten file will be copied and renamed as a puppy .

cp -r [dir1] [dir2]: This will copy the content that is within directory 1 and place it into directory 2. As before, if directory 2 does not exist, it will be created, and if it does exist, it will override the other format.

echo: This will trigger the system to send an input string to standard output. For example, if you are typing input, you can cause the input to be recorded and repeated in the output display.

find: This will allow you to do a search in the file directory for anything you are looking for.

grep: This will allow you to search for input files within a specific pattern and bring back the lines related to the search.

head: This will trigger the system to print the first 10 lines of the file for you to view in the terminal. So, if you use the following code:

```
$ head kitten.txt
```

then you would find yourself with the first 10 lines of the kitten poem in the terminal .

less: This command displays the contents of any named file one page at a time.

locate: This can also be used to find a file that you are looking for if you know its name.

ls: This command triggers the terminal to list out the content within the specified directory being used.

ls -la: This command will list out all of the directory's files, including any hidden directories that may be within it.

mdeltree: This is used to delete MS-DOS files, recursively deleting them and their contents.

mkdir: This command creates a new directory if the name you have entered does not exist. For example, if you wish to create a directory, you may type:

```
$ mkdir newfile
```

So long as you did not already have a directory named “newfile,” one would be created for you .

mkdir -p: This adds a bit more customizability to mkdir. With this particular command, you are able to also create subdirectories nested within the one you have named.

For example:

```
$ mkdir -p newfile/pictures/puppy
```

This would create a directory known as “newfile.” Within the newfile, there is a directory formed that is named “pictures.” Inside of pictures, there is a directory named “puppy.”

more: This allows for the display of content within a file page by page.

mv: This allows for the renaming of files and directories. For example, if you wish to rename kitten.txt into puppy txt, you would use:

```
$ mv kitten.txt puppy.txt
```

This particular code can also be used slightly differently in order to trigger the file to be moved from one place to another. For example, let’s say we have a directory named file and directory named basket and kitten.txt is currently within the directory named file. However, you want to move kitten.txt to the basket.

```
$ mv /file/kitten.txt /basket/
```

This code should facilitate the move, and kitten.txt will now be found in directory basket.

pwd: This command allows you to see exactly which working directory you are in for easy reference.

rm: This causes the deletion of a file that you have dictated. For example, if you wish to delete a text file named kitten, you would use the following code:

```
$ rm kitten.txt
```

rmdir: This triggers the system to remove or delete the specified directory, so long as the directory is empty at the time.

rm -f : This command forces the delete of a file, even if the file would otherwise resist it .

rm -r directory: This will delete a directory, its copies, and all of its content.

rm -rf directory: This will forcefully delete a directory, its copies, and all of the content within it.

tail: This command prints the last 10 lines of a file. For example, if you use the following code:

```
$ tail kitten.txt
```

then, you will see the final 10 lines of the fluffy kitten poem.

tail -f: This will provide the last 10 lines of a file, and continue to display the last 10 lines, even as new ones are added. In this instance, if you were writing your kitten poem actively in the file and used the code:

```
$ tail -f kitten.txt
```

then, you will find that every new line that you add at the end of the file will display itself in the terminal as the last 10 lines continuously refresh themselves. This is particularly useful if you are using it to check a live activity log so you can constantly see the most recent activity.

touch: This command tells you to create a new file. For example, if you wish to create a text file named kitten, you would use the following code:

```
$ touch kitten.txt
```

whereis: This file allows for the binary, source, and man page files for any given command to be located .

Users and Groups Commands

Linux can accommodate multiple users of the operating system, and those users can sometimes be combined into what is known as a group. Effectively, then, the group is nothing but a collection of users that have the same level of access privileges to the OS. Of course, one particular user can be sorted into several of the different groups at the same time in order to grant several different accesses and privileges if necessary.

groups: This command, when provided with a username, will provide you with all of that particular user's groups, and therefore allows you to see his or her access to the system. This is formatted as:

```
$ groups dad
```

This would then list out all of the privileges of the user on that system that was named "dad."

When you forego the username, it will display all of the information for the current user that is accessing the terminal .

groupadd: This will allow for the addition of a new group through the format:

```
$ groupadd mom groupnamehere
```

With this format, a new group named "mom" has been created.

groupdel: This allows for a specified group to be removed.

groupmod: This allows for specific group definitions and permissions to be edited and modified.

gpasswd: This will allow the admin to alter the group passwords to gain access to the system.

passwd: This allows for the changing of a password for the user activating it.

users: Allows for the current list of active users on the machine to be seen

useradd: This allows for the creation of a new user for the syste m

userdel: This allows for the removal of a user account and all files associated with it

usermod: This allows for a user account to be modified .

Files Permissions Commands

This section will provide you with all of the information you need in order to change directory permissions. In doing this, you are able to make larger level changes at a group level rather than having to manually change an individual's permissions. When making changes here, you will primarily see three specific letters repeated: r, w, and x. These stand for reading, write, and execute respectively. When you see these added to one of these groups, you can infer that these privileges have been added for a specific file or directory, or if you see them subtracted away from a group, you can infer that they are being revoked.

chgrp: This refers to the ability to change the groups of files and directories. When you use the commands that have the chgrp root, you know that you are messing with the groups of files. In particular, you must keep in mind that all of the groups must have their accounts logged out to assign groups. There are two specific commands for this particular action .

chgrp groupname filename: This will allow you to change the groups of files to a new file.

Chgrp groupname foldername: This allows you to move the groups of files to a specific folder.

chmod: This refers to changing directory permissions.

chmod +rwx filename: this will add permissions to read, write, and execute.

chmod -rxw directoryname: This will revoke permissions to read, write, and execute within that directory.

chmod +x filename: This will allow for executable privileges to the dictated file

chmod –wx filename: This will allow for the permission to write and execute the file to be revoked.

chown: The last of the ownership commands that will be discussed is chown. This will help you change ownership of any given files .
This can allow you to move ownership of a specific folder to another person altogether. Primarily, this is done in two ways.

chown username filename: This will transfer the ownership of the named file to the named user

chown username foldername: This will transfer the ownership of the named folder to the named user .

Archives and Compressed Files Commands

Sometimes, what you need to do with your OS is to create archives or compress files. That is what the following commands are good for—they enable you to be able to take a large number of files and create an archive of them. They may also compress the archive in order to save space, depending on the commands you insert.

gzip: This is used to compress documents into a zip file.
rar: This will create and manage a RAR file in Linux.

tar: This is the GNU version of tar archives. When you use this code, you are able to store several files in a single archive.

unzip: This will tell you whether you can unzip the file, providing you a usage summary of the document. From there, you can choose to extract the file wwith gunzip and then open the file with tar xvf.

unrar: if installed on the system, this will unarchive and unpack the files that they are commanded to.

Chapter 5 Quiz

Congratulations! You have made it to the end of Chapter 5! At this point, you should at least have a general understanding of the several different commands that you can use to control your terminal. Try to answer the next five questions to determine how well you are currently understanding and comprehending the material before moving on.

1. **What is the sign that the terminal is ready for your input when using bash?**
    - **a.** !
    - **b.** @
    - **c.** $
    - **d.** %

2. **What does "x" stand for in permissions?**
    - **a.** Exit
    - **b.** Expel
    - **c.** Expedite
    - **d.** Execute
    - **e.** Excel

3. **Are uppercase letters acceptable to use?**
    - **a.** Yes, the computer reads them all the same.
    - **b.** No, the computer differentiates between the two and will not read an x as the same as X
    - **c.** Sometimes

4. **True or false: When you are programming, the order of words and commands does not matter.**

5. **True or false: If you are looking for a specific file, you will have to search manually.**

Chapter 5 Answer Key

1. **C:** bash uses $ to communicate that it is ready to receive input. csh uses %.
2. **D:** x stands for the ability a user or group has to run or execute the program or file.
3. **B:** While computers may be able to differentiate and translate now, the OS is still designed to receive all input via lowercase.
4. **False:** Programming is a language and has a very specific syntax that you must follow. Otherwise, you will start mixing up files and sending things to the wrong place.
5. **False:** There are several commands that are present to help you locate the file you are in need of.

Chapter 6: Control Privileged User

At this point, it is time to begin discussing the security and privileges of users that are able to access your OS or server. Especially today, people seem to think that technology and servers are less safe than ever, but contrary to popular believe, there is actually plenty of support available to keep systems safe and secure. One such way is through controlling privileged users, which this chapter will teach you to do.

In reading this chapter, you will be guided through various types of linux accounts, such as the root user, normal user, and system user. You will discover sudo, which stands for the super user do. This is the command for the system admin—it is the one command that allows access to everything and anything, so long as you are on the right account with the proper privileges to use it. Lastly, you will be walked through the sudoers file. This file is essentially the underlying programming behind the sudo—it controls the sudo and who has access to the sudo command in the first place .

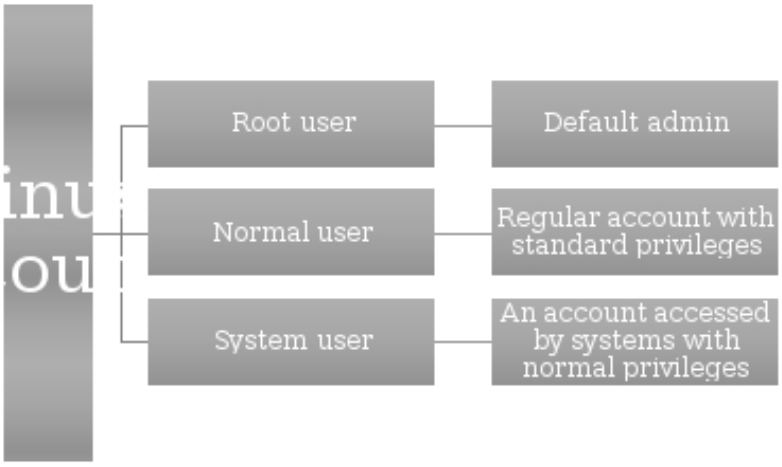
All of this will help you juggle the users on your server or machine, allowing you to ensure that the proper security systems are in place. With everything in place and a secure system, you are able to rest easy and feel like your system is truly safe .

Types of Linux Accounts

First, we will be discussing the root, normal, and system users before progressing to anything beyond this. You must have a complete understanding of these three user account types in order to really understand how sudo and sudoers play into the security of the system.
Because Linux is so incredibly flexible, you have the option to create several different types of accounts. In fact, you can have several root accounts if you need them, or accounts with different permissions in different categories. All that matters is that the needs you have are met effectively.

Keep in mind that just because you have root accounts does not mean that you should not have some sort of security system in place—you should absolutely have a password protecting the root accounts to ensure that no

one else is ever able to access those files and overtake them without your permission at all. This means that you will be able to ensure that the files themselves have yet another layer of protection. Now, without further ado, let's begin to dig into the three primary user types of Linux.



Root user

The first user that you must know is the root user. This is also commonly referred to as the super user, especially in the acronym sudo. The super user or root user is the individual that is on the default admin account. This person has full control over the system, much like how an administrative account would have control over Windows. With the root privileges that are given by default, this user is able to execute any of his or her commands and directly control or alter the services and accounts of other people.

Effectively, this should be your account if you are the one setting up the system. This account is the one that is able to rule over the other accounts, managing and monitoring them all without need for extra permissions. All permissions are granted for the root user. They are able to change the permissions of other users, move files around, change the ownership of files, and do anything else within the system.

Keep in mind that, because you have the power to do literally anything within the program, you will not have the OS telling you not to do something. There is no safeguard for you, and you *must* be careful when using this account. If you delete a system file, there may be no hope for going back if you have not been regularly keeping archives and backups of

your file. If there is a user that is capable of doing catastrophic damage to your program, it is the super user or root user.

Linux assumes that the root user has a pretty good grasp over what they are doing, and because of that, if you are unsure of what you are doing at any point in time, it is better safe than sorry, and you should try reaching out to someone else for help if you think it is necessary. Overall, while this should be your account with your own privileges, you should also try to avoid this account as much as humanly possible in order to ensure that you are not risking destroying everything at all.

There are very specific periods in which you *should* use the root. Otherwise, you are likely going to be fine with just a normal account for most day-to-day functions. Of course, any time that you are attempting a task that requires you to a user the root, you should make sure that you switch over as necessary and back up your data before attempting anything. In particular, you will need to use the root privileges in order to move files or directories in or out of the system directories. Remember, the system directories are those that are directly responsible for the functioning of the operating system, and without them, you cannot run the OS. It is also necessary when you are attempting to copy any of the files into system directories, when altering the user privileges of other users, for some system repairs, and the occasional installation of a program will all require you to use the root account. Typically, if a file needs root permission to be installed, it is because the file itself will be directly interacting with the system directories. If you are not doing any of the above, just avoid it and save yourself the hassle of potentially messing up.

Normal user

The normal and system users have the same privileges, and both are typically created by the root or another user who has been granted some level of sudo privileges .The normal user, however, is normally a real person, while the system user is typically a user reserved for software to utilize if necessary.

The normal user will have a real login shell and its own home directory, and each user will also be assigned a UID—a user identification. If this went

unspecified during the creation of the new user with the **useradd** command, then the UID is typically automatically selected for you.

Essentially, however, this user does not have any super special admin powers or superuser status. The normal user is able to access any files that have not been otherwise marked as unavailable, and the privileges of the normal fire can be changed regularly depending on what is necessary for the user at any given time .

Most likely, if you are running Linux on your own device, you will have a root account, the administrative account that was used to set up the entire OS, to begin with, and then you will also set yourself up with a normal user account to ensure that you do not accidentally delete something critical or destroy something important. This is your sort of safeguard against an accident.

The normal user's own permissions can be customized by the root user—you can grant yourself any permissions that you desire on the normal account from the root account, so long as you know the right way to code it.

System user

The system user, as briefly touched upon, is granted the same initial privileges as the normal user. However, the system user is almost always some sort of software or program, such as a daemon, that will be running in the background and will likely not be particularly interactive. This user classification primarily exists just for organizational purposes. When you are able to dictate the account as a system account, you know at a glance that it is not going to involve any sort of people interacting with it. There may be a daemon running the user, but no person will ever be longed on.

While in the real world, this may not necessarily matter technically, it is nice to be able to see at a glance how many human users you have as opposed to daemons running specific software. Usually, these system accounts are created by the operating system during installation, and the OS retains their control. The usually have very specific user ids that you can identify within the root account.

Because these service users will be running processes necessary for the OS, they should be left alone when you are altering, ending, or resetting any processes. You want to make sure you leave these service users alone as much as possible to protect yourself from damaging the software. If you were to damage one of these, it is quite possible that you would destroy the entire OS, which is unfortunately easy to do if you are not paying attention .

Sudo

Now, it is time to discuss sudo. If you are using sudo before a command, you essentially push it up. You tell the system that you are commanding this particular process with elevated privileges that should be respected. Usually, the elevated privileges are what the system checks to ensure that you are actually granted the privilege to make any alterations that you are actually making.

If you have been using Windows primarily, you already have interaction with a sort of example of this account. When you install software, have you noticed how certain installations will trigger a prompt to pop up and ask if you are certain that you wish to proceed and ask for you to give the software the permission it needs to proceed? That is what you are doing in Sudo. You are effectively giving that permission, but instead of with a click of the button, you are telling the system that you have elevated access to the system and should be respected .

If you have been primarily using macOS until now, you have a similar sort of safeguard when you trigger downloads that may be recognized as dangerous or will create major alterations to the OS. On a Mac, you will see a security box pop up, and you must then provide the system with your password to confirm that you do, in fact, want to perform that change on the system.

Effectively, then, Sudo is your privilege pass. It is the magic passphrase that tells your system that you are in charge—so long as the privileges on your user account match.

Using Sudo

When you are ready to use Sudo, you do not have to try very hard. Effectively, all you have to do is add **sudo** before the command that you are trying to push forward. If you do this, your program should then ask you for the account password for the account that you are on. If you provide this password and your credentials check out, it will perform the function .

For example, imagine that you want to reboot your system. As you may recall, the command to trigger your system to reboot is simply **reboot.**

However, reboot requires sudo privileges. If you were to simply try commanding your system with just reboot, it would tell you that you must be a superuser in order to execute that particular command.

That is where you would then escalate your command. At that point, you can add to your command, creating an input of:

```
$ sudo reboot
```

Your system then asks for the account password, and upon entering the password, it reboots as requested.

Su instead of sudo?

This is perhaps one of the safest ways that you can use to elevate your privilege to trigger the system to obey. There are others, such as the switch user command, known as "su." However, when you use the switch user command, you are asked for the root password and then given the superuser prompt.

You know that you are in the superuser prompt when you get a # as your input ready signal instead of the typical $ that you are likely used to seeing by now. The # should be your warning sign—if you see this waiting for you in the terminal, you know that you are on the root account, and you need to be careful.

While the switch user command may have its uses in very specific situations, it also runs the risk of you destroying everything with a single typo or the act of someone accidentally bumping you while you are

clacking away at your keyboard. When you use sudo, however, you are required to insert it for each and every command that requires extra privileges. This means that you have that extra safeguard. Of course, this could get annoying if you were in the process of doing something that would require you to constantly be validating your credentials, but it could be worthwhile if you think that the extra security is worth it. Of course, you could always just set up a backup of your system right before you begin the process of entering your su commands if you wanted to bypass the constant sudo usage, but that is ultimately a personal decision that you have to make, weighing the risk for yourself. Ultimately, the more you are in su, the more likely it is that you are going to have an accident .

The Sudoers File

Underneath sudo is the sudoers file. This determines who gets to use the sudo command in the first place in order to actually make use of the authority it brings with it within the OS. Typically, you can access this file within the location of /etc/sudoers, and you will have to edit this if you want to mess with command values and permissions. Perhaps the best way that you can make user of the sudoers file and the safewst way to editing it is thorugh using the visudo editor.

Visudo

Visudo itself acts as a sort of stopgap—an extra layer of security for the sudoers file. It allows for safe edits while also locking the sudoers file from being edited at the same time by multiple users, which could become dangerous if two people tried to create contrasting edits at the same time. If you try to access the sudoers file with visudo when someone else is currently in it, you will be rejected with an error message and told to try again later. When you do make an edit, it will stop and check the edits for any errors that may be catastrophic in an attempt to at least try to make sure that you do not completely destroy something important .

Upon finishing your edit of the sudoer file after accessing the visudo editing system, the edit will be scanned for any errors. If you do happen to make a

syntax error, it will reject the save, printing the message that declares that there is an error and will tell you which lines that mistake is in. From there, you are prompted to either attempt to re-edit to fix the mistake, or you can quit saving the changes anyway. However, if you quit after visudo has found an error, it is highly likely that sudo will also find the error, and sudo will not be accessible until the error has been corrected.

In order to access visudo, all you have to do is enter the prompt:

```
$ visudo
```

and you should be granted the access you are looking for, so long as the permissions are right. There are also several other options that you can pair with the visudo function in order to help you really make the most out of the program. These commands include :

visudo -c: This triggers visudo to go into a check-only mode. It will check the current sudoers file for any syntax issues and will print the status of the file. If the check was clear, visudo would end with a final value of **0.** However, if there is any sort of error detected, it will end with a value of **1.**

visudo -f sudoers : This specifies an alteranate location for sudoers, checking or editing the file of your choice instead of storing it in the default /etc/sudoers.

visudo -h: This is the help option, and when you do this, visuo will provide a short printed message.

visudo -q: This tells visudo to enter quiet mode. When in this mode, no details of the syntax errors detected during a search will be printed or pointed out to you.

visudo -s: This will enable the checking of the sudoers file. It will create an error if an alias that has not been defined is being used.

visudo -v: This is the version option—it tells visudo to print out the version number before quitting.

The sudoers fil e

With the visudo access created, you are then able to begin editing your sudoers file. This is what will provide the sudo access to any accounts.

When you first access your sudoers file and scroll to the bottom, you will likely see a line that says something along the lines of:

root ALL=(ALL) ALL

Effectively, this means that the root user can execute from all terminals, like all users, and run all commands. Read all as any for a moment, and you can see how that can be scary. The root user can be basically anything it wants to order anything it wants and do anything it wants.

If you want to have any similar powers yourself on other accounts, you will need to set yourself up for an account that is designed as a sudo account.

First, you must log into your server as the root user.

Then, using the command **adduser** you must create a new username.

$ adduser usernamehere

At this point, you will be asked to make a new password for the account. Do not forget your password, and make sure it is actually secure. From there, the command will create a home directory for you. Just press Enter to accept the default unless you feel like filling it in.

Then, you must add the new user to the sudo group using the following command

$ usermod -aG sudo usernamehere

This should give you the sudo access, though you should still probably test it. Switch to the new user with the **su** command:

$ su – usernamehere

At this point, you need to test your access. Try using the whoami command to get all of your details:

$ sudo whoami

If done properly, the command will provide the output “root.” If it is root, you know that you have access.

From here, all you have to do is use the sudo command with space after it before any command you issue. The first time you attempt to use sudo when in a session, you will be asked to enter your password for security reasons .

And just like that, your user account will have sudo access, allowing you to avoid any painful mistakes that could cripple your system or ruin your files in any given way. Since you will now have to enter sudo in order to do anything significant or dangerous, you should be pretty secure.

Chapter 6 Quiz

Congratulations! You have finished reading Chapter 6: Control Privileged User. By now, you should have a general idea of how the system hierarchy of users works within Linux and how you can access the right level of security for yourself. At this point, you will be provided with five questions, as usual. Try to answer these questions to determine how well you have comprehended the material presented thus far. As always, the answer key will be provided on the page after this quiz.

1. **Which user type has the most privileges?**
    - **a.** Normal user
    - **b.** Root user
    - **c.** System user
    - **d.** Random user

2. **What is the command necessary to override administrative blocks without having to log into the administrative account?**
    - **a.** sudore
    - **b.** visudo
    - **c.** sudo
    - **d.** su

3. **Which statement is true?**
    - **a.** You should avoid the root user account due to risk of deleting important information
    - **b.** You should avoid the use of visudo editing due to the risk of deleting important information
    - **c.** You should avoid the use of sudo commands due to the risk of deleting important information

d. You should avoid using Linux due to the risk of deleting important information

e. All of the above

4. **True or false: Visudo will check for errors when you make edits.**

5. **True or false: Visudo will publish a faulty edit to the sudore file without warning by default.**

Chapter 6 Answer Key

1. **B:** The root user has all privileges by default.
2. **C:** sudo is the command used to override administrative blocks
3. **A:** You should avoid using the su root account if at all possible due to it lacking the safeguards of all of the other mentioned processes.
4. **True**
5. **False:** You must command it to save the faulty edit manually, overriding it from warning you about the presence of a syntax error to begin with.

Chapter 7: Basic Network Administration

Thanks to just how flexible and customizable Linux is, it is incredibly versatile in the world of network administration. This makes it particularly useful if you need to run a data center or some sort of server. This chapter will provide you with the last of the information you need to know to get started with your Linux setup. You will be guided through networking as if it were all new to you—because as a beginner reading this book, it may very well be!

Within this chapter, you will learn about the network extension and topology, learning how it works. You will be guided through several protocols, given some information on routing, given some commands that may be useful to you, and you will be on your way. The following chapter will walk you through deciding upon some various software for a Linux distro install, but this is the final chapter with all of the hard, dense material. Congratulations—You are almost to the end!

Networking 10 1

Networking allows for the connection of computers, phones, peripherals, IoT devices, and more to all connect together, accessing one line of data and being able to communicate. This is the idea with the internet—you are able to access the internet network and communicate with devices that are even on the opposite side of the world from you. There are several important points to consider with networking, but at the base, think of the internet as one giant web—it is all interconnected with several access points for other devices to get to. No matter where you are on that web at that moment, you can get to any other point on the web in some way, even if that way may be slower than it needs to be or should be. This means then that you are able to sort of jump your way around the network to access data that you need, even if it is not on your physical machine. That is how the internet works.

Networking has far more uses than just general web browsing, however. It can be useful for the distribution of data, data storage, general networking, and more. Thanks to this web, we are able to stream videos with ease from anywhere we are able to connect to the internet, or able to access files on

the cloud. All that you need is a steady connection, and you are able to access nearly anything that is also connected .

The Network Extension

Networking spreads out across several people, allowing for rapid distribution of data if necessary. Ultimately, the network can be thought of as being divided into three distinct categories: The LAN, MAN, and WAN. As depicted below, you can see that the Lan is the smallest, with the MAN being larger, and the WAN is the largest form of network. Each subsequent network type is going to be comprised of several other aspects of the previous. Essentially, several LANs create a MAN, and several MANs create the WAN.

WAN

Standing for Wide Area Network, the WAN is the communication network across any given specified geographical are. It may involve a city or even a country, and these may be public or privatized depending on the usage and whoever is managing it.

To understand what the WAN does, stop and think about the internet—now imagine this as the biggest WAN there is. Several networks are connected together to create one big process. This is what a WAN is—in a WAN, you will have several smaller LANs or MANs that are all within one area, making up the WAN. The WAN is widely connected and is able to reach farther than a LAN. This makes a WAN the best choice if you wish to run a network or a wide-scale sort of server .

LAN

Standing for a Local Area Network, the LAN is much smaller. Usually, the LAN covers a single building, such as a house, or even just a single office in a building. It is meant to be much smaller and not accessed by nearly as many people. They are most frequently connected via Ethernet to control the dataflow, though more and more LANs are moving toward wireless, known as Wireless Local Area Networks or WLANs.

If you have a home setup in which you have two systems sharing access to data, then you have a LAN. Other systems beyond computers can also comprise the LAN, such as printers or smart televisions that will share the data. These are used for the sharing of local data as opposed to wide-scale distribution. However, they are incredibly useful for a home or for a small business .

MAN

Standing for a Metropolitan Area Network, this is larger than a LAN but still smaller than a WAN. This may be, for example, a large tech campus that shares a network spanning several blocks or even several square miles, but is not particularly wide-reaching—this would be a MAN. It can also be used to describe a city. Typically, a MAN is several LANs that create it, and the MAN is a fantastic way to pass forward the data within a LAN, allowing it to stretch further .

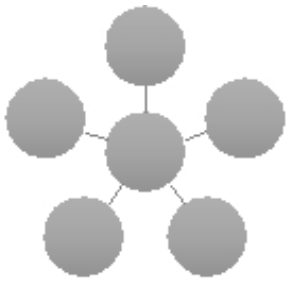
The Network Topology

With that understanding of the network itself, it is time to look at the most common topologies—the forms through which networks connect to nodes. The node is an individual device that is connecting to the network in some way, and different topologies will organize these networks in different ways, depending on necessity and what makes the most sense.

Point to point

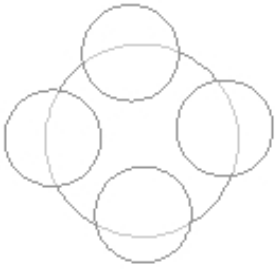
With this network topology, sometimes referred to as the bus topology, there is a central cable that runs between all devices on a LAN and connects them all. It is easy to connect to, but if there is any sort of break in the system, the whole thing will crash.

Star



The star topology works to connect several devices all to a central computer that is referred to as the hub, as seen in the picture above. The nodes are able to communicate by first passing through the hub. In this form, a single node's malfunction will not impact the rest of the network, but at the same time, if the hub were to fail, the entire network would be crippled.

Ring



In the ring topology, a LAN is set up with the topology of a ring. Each node is connected to two others within a closed loop, and all messages will travel around that circle. This allows the message that is being passed along to be regenerated with each node it goes through.

Tree

The tree topology is a hybrid of the bus and star topologies to create what looks like trees when drawn out. It involves several star-configured networks that connect to a single linear backbone. This is great for larger computer networks that are benefitted by being broken into easily managed pieces, but the system is at a disadvantage because the backbone acts as a hub, and if that fails, everything fails.

Mes h

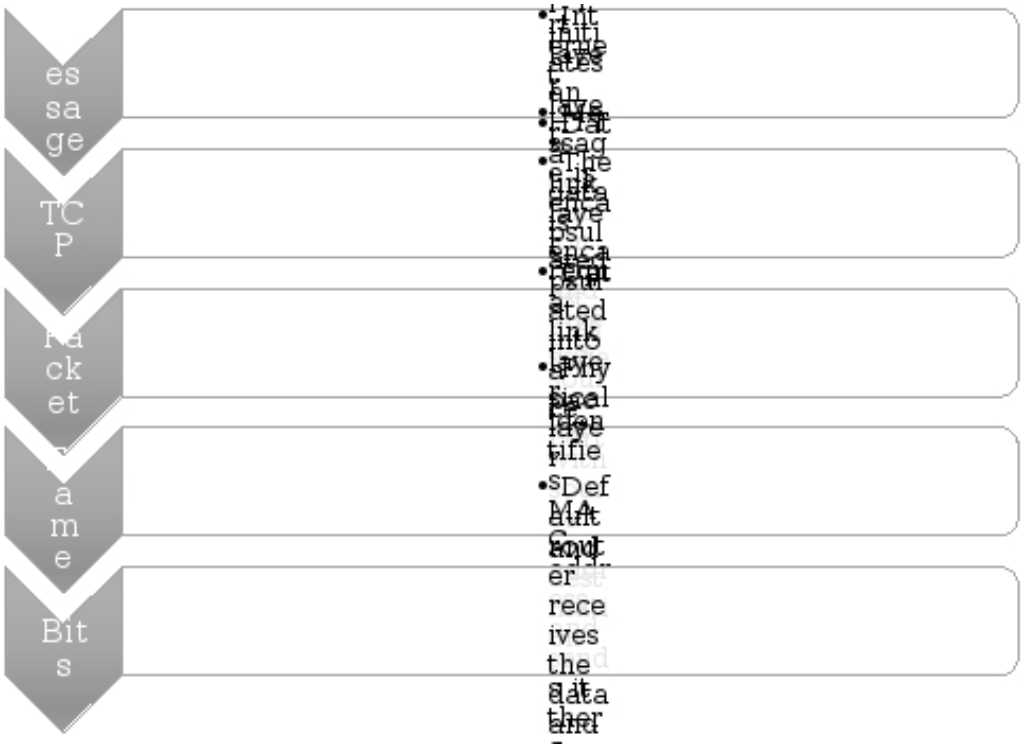
Full mesh topology connects every node to a circuit connecting it to every other node within the network. It is expensive to manage, but it also leads to the most redundancy—which means that it is safer. Even if a single node fails, traffic can continue because all nodes are interconnected.

Sometimes, some nodes are organized with a full-mesh pattern while others are not entirely interconnected, allowing for some redundancy and some money saved, while still managing an effective network. This is known as partial mesh topology.

Main Protocols of the Interne t

TCP/IP

As a precursor to routing, you must first understand the TCP/IP network model. This is the way that packets are able to find their way back and forth across the internet, allowing them to reach the right host within the network. Within this network model, there are five layers that describe the necessary processes to actively shift and move the packets from the host to host as necessary, no matter whether that host is local or across the world.



- **Application layer (Message):** It all starts with a message. The first layer consists of the necessary protocols that will be needed for communication, such as HTTP, SSH, IMAP, and any other commands that will be necessary. When you then request a webpage at any point, the message is communicated through one of those formats.
- **Transport layer (TCP):** Next is the TCP segment—this provides the data transport from end to end and also manages the flow. It is independent of the data that is being transported.
- **Internet layer (Packet):** This is done on the internet layer of things. It is the part that routes data across the appropriate networks in order to ensure that they arrive at the proper final destination through the use of IP addresses and routing tables.
- **Data link layer (Frame):** This layer controls and facilitates the direct connections that occur between hardware within a single physical network. It uses MAC (Media Access Control) address within NICs (Network Interface Cards) in order to recognize any devices that are connected to the network. It is only able to

interact with other devices and hosts that share a network together.

- **Physical layer (Bits):** This is the layer consisting of NICS and a physical Ethernet cable that allows for the connection between networks. The Ethernet cable is what enables the bits to be transmitted.

IPv4

This refers to the Internet Protocol Address version 4—it is your IP address. With the IPv4, you are using the 4th protocol. Effectively, the IPv4 address will be a 32-bit number displayed in dotted decimal notation. You will see two primary components—the network's prefix and the host number. Every host sharing the same network will also share the same network address. However, each host will also have its own part of the number that is unique. If the device is accessing the internet and is visible to those outside of the network, it must contain a globally unique number.

TCP

Standing for the transmission control protocol, this is one of the main protocols of the internet protocol suite. It was designed to create a standard that allows everyone to establish and maintain a network conversation because everyone is using data that is compatible. It allows for the exchange of data in the form of packets .

UDP

Referring to the User Datagram Protocol, this is an alternate communications protocol to the TCP model discussed above. The UDP is used to establish low-latency and loss-tolerant connections to allow for high speeds and low risk .

Routing

Now, at this point, you may be considering turning your Linux server into a router—this is a valid point, and several people do it. In order to do this, you must first do a configuration of the router, and then you must assign the address necessary.

Linux as a Router

First, you must configure the device. This will require you to set an IP address on two devices first, making sure that they each use the same gateway IP. Now, what you want to do is set your network to accommodate that gateway IP.

To do this, you must first login to the root account, and then run the command

 # nmtui

Next, choose the option to **Edit a connection** and hit the **Enter** key.

Choose the available Ethernet that is present on the leftmost side of the window and select **Edit** on the right, once again pressing **Enter .**

At this stage, you will have the opportunity to set your IPv4 configuration to manual. Then, you need to enter the IP addresses that you wish to have access to the network. You can add more if you need to.

Select **Quit** and then press **Enter.** At this point, you should restart the network, and you will need to then confirm that the changes took. In order to do this, you will enter the command:

#ip addr

You should ten be able to see the systems that can access the gateway without accessing the other network. You can confirm this with a ping command. The ping should come back as the destination host unreachable.

Ifconfig command

ifconfig is a CLI designed for network interface configuration. Beyond just that, it also serves to initialize interfaces during boot up. When the server is actively running, ifconfig can further be used to assign IP Addresses to interfaces, as well as enabling or disabling any interaces on demand .

This command will also allow you to view the status IP, any hardware and MAC addresses connected, and the maximum transmission unit (MTU) for all active interfaces.

Overall, this system command is designed to allow you to debug or to tune the system.

DNS settings

If you find that your DNS settings are misconfigured or you need to change them in some way, there is a simple enough process for this. You will do so by adding name servers to your configuration file.

Within most Linux systems, the DNS will create names defined within the **/etc/resolv.conf** file. This file will usually contain at least one nameserver line. That line will be the defining line for the DNS server. The name servers will be prioritized by the order that they are found within the system. For ease of access, use the IP addresses of the servers you will be using.

To manage this, you will need to open the resolv.conf file with some sort of editor program, such as nano. In doing so, you will be able to make the changes. If nano is not yet in use on your system, try the following command to install it:

```
$ sudo nano /etc/resolve.conf
```

You should then have access to add lines for the name servers you are using. Most often, you will want the one that will be housing any cloud servers you have.

With the line renamed, you can then save the file.

Of course, as with after any major change, check the ping to ensure that the domain settings took and confirm that you are, in fact, using the specified IP address. If you have gotten a message at this point that says that your host was unknown, you may need to try resetting the IP address one more time, as there may be an error within the one that you set up.

/etc/hosts

This command will allow you to configure the DNS locally instead of using external IP addresses. When you use the **/etc/hosts** command, you will

ensure that the naming structure is right. It will allow you to manually enter in the IP address and hosts of your choice to confirm that you are able to get the domain name you were looking for .

Diagnostic commands

ping: This is how connectivity is tested. Standing for Packet Internet Groper, ping is regularly used to identify that connectivity between either LAN or WAN. It will make use of ICMP in order to communicate with other nodes on the network.

traceroute: This is a CLI that allows for the tracing of the entire path from a local system to a separate system on another network. It will record the number of hops between router IPs that will have to occur to reach the server necessary.

tracepath: This command allows for the path to be traced to the destination discovering MTU. It will utilize a UDP port. Similar to traceroute, but this particular form of tracing does not require the same superuser privileges because it has no real options. It is simply informational and will provide all the necessary information .

netstat: This CLI is used to display the information you need to know about your network connections, any routing tables, statistics, and more that will be important to understand with your networking system. It is also used to debug and check to determine which programs are on which ports.

Chapter 7 Quiz

Congratulations! You have finished Chapter 7! You should now have a general idea of how networking on your Linux machine will work and key features that you will encounter. As with previous chapters, it is time for your 5 question quiz. This will help ensure that you understand what has been presented within this chapter.

1. **What is a network?**
    - **a.** A group of people that you know
    - **b.** A group of devices that are connected together
    - **c.** A group of software that communicates with each other
    - **d.** A group of neighbors that all share the same internet provider
2. **What is WAN?**
    - **a.** The internet
    - **b.** Wide area network
    - **c.** Widely accessible network
    - **d.** Wonderful access network
3. **What happens if the hub of a network goes out?**
    - **a.** The whole network stops working
    - **b.** Half of the network stops working
    - **c.** Only the hub stops working
    - **d.** Hubs are obsolete and no longer used
4. **Why create a network in the first place?**
    - **a.** Easy distribution of files
    - **b.** Sharing data
    - **c.** Connectivity
    - **d.** Redundancy
    - **e.** All of the above
    - **f.** None of the above

5. **True or false: You really only need access to a network if you are going to be gaming or otherwise streaming large amounts of data over the internet**

Chapter 7 Answer Key

1. **B:** While you do have a personal network, that is the wrong context for this chapter. The network in the sense of Linux is a group of connected devices that are able to share files in some way.
2. **B:** Wide area network. While the internet can be considered a WAN in some sense, it is not what WAN stands for. This is a large geographical area of networks.
3. **A:** The whole thing will stop working. When a hub is implemented, it is the central processing power for the entire network, and without the hub, the rest of the system has no real way to run properly.
4. **E:** All of the above. There are endless reasons that setting up a network could be useful if you have a legitimate need for one in some way. They are absolutely recommended if you need to regularly share files or you want to easily shift from system to system within a home.
5. **False:** In fact, networks can actually work incredibly well as long-term storage for files, which is letting files rest, unused rather than using intensive internet processes that gaming or streaming would involve.

Chapter 8: Alternatives to Windows Applications

If you have made it this far in the book, then you are likely either seriously considering switching to Linux, or you have already decided to do so. Either way, you may have some reservations about the process. For example, it can be somewhat daunting to stare into the face of an operating system that is not designed to have all of your proprietary software. Especially if you have been a long-term user of Windows OS, you are likely going to feel like you do not want to miss out on some of those programs that have been implemented into daily life. After all, you may need access to Microsoft Office in some instances, such as for work, school, or just due to your own personal preference. While running a virtual machine to run Microsoft just long enough to get all of the files you want tis absolutely an option, it is likely not the option that you really want to go with .

Luckily for you, there are several substitutes out there for the mainstream Windows applications, and these substitutes even come with one more bonus—they are oftentimes free. While this may cause you to hesitate at first, wondering if the free software is going to be less reliable or less complete than what you are used to, consider that Linux itself is a free iteration of an OS and you are likely quite close to using that at this point in time.

Despite the fact that these projects are free, they are still useful. In particular, this chapter is dedicated to finding you the best popular alternatives to some of the most mainstream Windows applications you will use. From word processing to PDF viewing and all the way to a substitute for digital editing programs, free options that are entirely compatible with Linux exist out there for you. This chapter will provide you substitutes for the following products:

- Microsoft Office
- Notepad
- Internet Explorer
- Photoshop

- Movie Maker
- Windows Media Center
- Adobe Reader

Of course, the options that are going to be recommended within this chapter are not your only ones—you can go out of your way to locate several other programs and applications that you may find to be more useful to you, and that is okay. Ultimately, using Linux is all about user satisfaction and customizability, so you should absolutely focus on designing it to be functional for you. Feel free to explore your further options to determine if you can find anything else that you would find to be particularly useful for you .

Microsoft Office Substitute

Many people utilize Microsoft Office on a regular basis. In fact, it is regularly used in school settings, with many universities even requiring that their students use this format for the submission of documents. Microsoft Office comes with several different programs within it, allowing for spreadsheets, word documents, and several other forms of documentation to be processed and compiled. If you have been in school recently, you have likely used this at least somewhat.

There are, thankfully, several other alternatives to Office, though you may run into instances where you are required to utilize Microsoft Office. Nevertheless, if you can get away with it, you can get by with **Libreoffice.**

Libreoffice is built to be free, and it is admittedly quite powerful. Within this suite, you are given several free and open source office resources. Like Office, it includes several different programs all bundled under one name, each of which produces a different type of document. With LibreOffice, you have access to several different programs:

- **Writer:** LibreOffice’s word processer
- **Calc:** LibreOffice’s spreadsheet processor
- **Base** : LibreOffice’s database processor
- **Impress:** LibreOffice’s presentation creator

- **Draw:** LibreOffice's diagram creator
- **Math:** LibreOffice's formula editor

As you can see, all of the major functions of Microsoft Office are provided there with LibreOffice, and for the price tag of free, it almost can't be beat, unless you really *need* to have word doc formatting. While you will be able to open up files created in Microsoft Office in LibreOffice, the converse is not true.

LibreOffice is frequently bundled in with many of the more common Linux distros, so you may not even have to go out of your way to locate this particular installation.

If you dislike LibreOffice, there are other alternatives as well. In particular, you may find that Open Office or AbiWord is more your style, and that is okay !

MS Notepad Substitute

MS Notepad is usually built in to Windows. It is commonly used as exactly what it sounds like—a notepad. It is a text entry system that you can use to store notes, writing, or anything else that you may have. However, it is far less sophisticated than Microsoft Office. Nevertheless, if you find that you miss MS Notepad, there are substitutes out there for you.

In particular, gedit is the GNOME text editor. It is designed to be simple and easy to use, while also allowing for the highlighting of several programming languages. It also comes with the undo/redo commands, clipboard support, and printing support. Basically, anything that you could possibly need in MS Notepad will be available to you with gedit.

There are also several other types of text editors that you may find to be useful, as well. For example, jEdit, Kate, and NEdit are all other free, Linux-based text editors that may be useful for you .

Internet Explorer Substitute

The vast majority of use on a computer these days is browsing the internet, and if you use Microsoft, you have possibly been using Internet Explorer as your browser. If you are one of those people who ditched Internet Explorer

as quickly as you could for something else, there is the chance that whatever it is that you were choosing as your browser is still supported. Internet Explorer itself is not compatible with Linux, but with so many other options available, you are bound to find something that you like.

In particular, Firefox is commonly recommended. Firefox is strongly recommended, even among Windows users, and for a good reason—it is incredibly customizable and offers support for several plugins. It is also actually more secure than Internet Explorer in the first place. Even better, Firefox is compatible with mobile devices as well, so if you prefer, you can set up your system so you are able to access all of your information across your computer and your phone when utilizing this system .

If Firefox is not for you, there are plenty of other browsers available on the market for you as well, and you should be able to find one that works well for you with relative ease. In particular, you may choose to look into Epiphany, Konqueror, or Opera .

Photoshop Substitute

Photoshop is the golden standard when it comes to photo editing software for many people, but unfortunately, it is locked behind quite a large paywall. Photoshop, while a good program in general, is quite expensive, and also more on the restrictive end. Photoshop is written in C++, making it rather restrictive in general, whereas you do have other options available to you.

In particular, you have access to GIMP. Unlike Photoshop, GIMP is written in C and GTK+, making it far more flexible to use. Even better, GIMP is far less strenuous on the processor while still giving you access to several tools. Not only is Photoshop monetarily expensive, it is also expensive on the processor, especially when it is editing files in high resolution.

GIMP typically gets bundled in with most common distros, though you may stumble upon one that does not include it. Nevertheless, try taking a look at it. Even if you do not make the shift over to Linux, you may find that GIMP is a better alternative to Photoshop for you anyway .

Beyond GIMP, if you find that it is not quite to your liking, you can try CinePaint as well. These programs, while being recommended for Linux, are also compatible with Windows .

Movie Maker Substitute

Despite common assumptions, Linux actually has several options available for decent video editing programs. While many people may feel like they are missing out if they do not have access to their media editing on Windows or Mac, such as Windows Movie Maker, you are actually able to find alternatives readily available on Linux as well. While Windows Movie Maker is a good option for you, if you have access to it, some of the lesser known Linux iterations are actually more advanced and allow for further functionality that you may not otherwise have access to.

In particular, Cinlerra is designed to be an advanced alternative to Movie Maker. In Cinlerra, you are able to work with ultra-high resolution processing of images. Despite being free, it offers you all of the editing software you could possibly need—and then some. It offers features such as color correcting, motion tracking, mastering audio, and more, all designed to give you everything you could possibly need when shifting over to Linux from Windows. After all, artistic pursuits should not have to suffer in the name of customizable operating systems. And even better, like all of the other programs that have been discussed in this chapter, it’s free.

If Cinlerra is not your style, you may find that one of several other options appeals more to you. Other options that you could look into alongside Cinlerra include Kdenlive, LiVES, Open Movie Editor, VideoLAN, and more. All you have to do is dig around a little bit, and you will find several other options for you .

Windows Media Center Substitute

Windows Media Center is the major entertainment system that was released with Windows starting with Windows 7. It allows you to load up videos. However, it was also rather strenuous on the graphics card, requiring a high-end one in order to actually function properly. Because of this, many Windows users ditched Windows Media Player long ago. Even if you were one of the common users of it, you are not losing out on much by transferring over to Linux.

Instead, try taking a look at Linuxmce instead. This program is not just limited to videos either—it can be utilized with several other functions and programs as well, such as security cameras, telecommunication, and also running all of the media that you may have had on your device to begin with. It is meant to be a sort of cohesive program that unites your electrical appliances at home, from lighting to security and media, with everything in between.

Since we are discussing this as a media specific replacement, we will focus on the media features. In particular, it is designed to be used to link together several devices on a network, allowing the files that you are accessing to be accessible in several different locations, which further allows you to ensure that you always have your favorite media available.

There are also several other options available for a substitute for Windows Media Player, such as Moovida, MythTV, and XBMC Media Center. Try playing around with several of these to find one that really works well for you .

Adobe Acrobat Reader Substitute

Especially in adulthood, Adobe Acrobat Reader is incredibly useful. Necessary to view portable document files (PDF), you must have access to some sort of reader within your operating system. Most commonly, when you use Microsoft, you will end up with Adobe Acrobat Reader. If you do have Adobe Acrobat Reader, you may also be familiar with the myriad of constant updates it puts out, even when little-to-nothing changes. This means that you are constantly being forced to update, even when it is unnecessary. Of course, if you are leaving behind Windows, you also leave behind this problem. Instead, you can use one of the free and open source software options that will eliminate that problem altogether.

Even better, many of these free options are readily distributed with the most common Linux distributions. If you are going to be running a common distribution, it will likely include either Evince, okular, or Xpdf, all of which are more than capable of meeting your software needs .

In fact, Xpdf, in particular, provides you with several different tools that will allow you to do more than just read your files. You will be able to

convert PDF to text, PostScript, into PPM/PGM/PBM image files, PNG files, HTML, and more. It can be used to extract metadata or raw images, and will even provide the fonts that were used to create the document for you. It is incredibly versatile, and while it does require you to understand the command line tools, for full optimization, you will find that it is more than capable of getting the job done.

Ultimately, as you can see, there are nearly endless opportunities for all of the major programs you are likely to encounter when using Windows on a regular basis. By just being willing to do the necessary research, you can begin to encounter a far more versatile operating system in general. While it may not necessarily be what you are useful, you do not have to spend an exorbitant amount of money just to get functional software that will do what you need it to. You can, in fact, get that service through the use of plenty of the freeware offered online. As a quick overview, let’s recap the most common replacements for the software discussed.


Microsoft Office
• LibreOffice
Notepad
• gedit
Internet Explorer
• Firefox
Photoshop
• GIMP
Movie Maker
• Cinlerra
Windows Media Center
• LinuxMCE
Adobe Reader
• Xpdf

Chapter 8 Quiz

Congratulations! You have made it through Chapter 8. By now, you should have a pretty solid idea of just how much software is readily available to you without having to pay insane amounts of money to access it. Try to answer these questions to ensure you understand the basics before moving on. The answer key will be on the page directly after this quiz.

1. **True or false: You can just use your old Microsoft software with Linux.**

2. **True or false: You cannot use the Linux software on Microsoft**

3. **True or false: These programs, both the free and open software and the proprietary software, are entirely able to communicate with each other**

4. **True or false: The Linux software is expensive**

5. **True or false: You will lose quality by transferring to Linux due to the lack of software available to you.**

Chapter 8 Answer Key

1. **False**
2. **False**
3. **False**
4. **False**
5. **False**

Conclusion

Well, that brings this book to a close. At this point, you have enough basic information to get started and begin to dabble in the usage of your own Linux distribution that you have chosen, installed, and began to program. Hopefully, you found the process of reading this book informative and useful to you as you made your way through it. The material discussed can be quite dense if you are not technologically inclined, or if you have never really dabbled in tech-related systems before, but it is worthwhile to learn.

The benefits that come from being able to alter and regulate your own operating system, whether from the ground-up if you were brave enough to attempt Gentoo or another advanced program, or even just running an open source distro such as Ubuntu that is designed to be easily accessible, you are giving yourself a whole new world of opportunity. You have the option to directly interact and control your computer. You can influence the processes your computer goes through. You can tell your computer to do essentially anything, so long as you are able to work out the code necessary for it. This coding is the foundation in many much larger projects. What starts as basic coding today can eventually become an AI. What begins as the first attempts at poking around your software can eventually lead to you developing your own OS. Of course, the end result will be what you are willing to put in.

One thing is for sure; however—when you begin to utilize Linux, you are developing a series of skills that are essential to learning.
It can be beneficial to know how to work with computers, and you may even decide to take this from a project to a hobby, and eventually even a career if it is something that has interested you enough to keep exploring. No matter what, however, what is important to remember is that you should make sure you stay up-to-date on your knowledge.

From here, you may be ready to install your own iteration of Linux, if you have not done so yet. If this is where you are at in your process, good luck! It is an exciting time when you are first beginning on this process, and you will surely enjoy it. If you are unsure whether you are ready to make the plunge into downloading a distribution for yourself, maybe you would find

interest in running a few different distros to test first, using the steps listed for you earlier in this book. You can play around with the system, learning which you prefer and which you would rather avoid altogether, which may help you make your decision sooner.

No matter what you decide, however, keep in mind that this was an intro to the subject. This book focused on providing you with the basic essentials to understanding what Linux is and how it works. From here, you may choose to research the specialized distribution you are interested in. You may begin to look into more of the uses that Linux offers and what you can do with the program. You may even decide to continue to books dedicated toward people who are at intermediate or even advanced levels to continue to grow your knowledge. No matter what you choose to do next, if you are willing to put in the time and energy you will find yourself successful in your endeavors. It may not be easy, but the end result will absolutely warrant the end result .

And lastly, if you have found this book to be useful to you in any way, shape, or form, please feel free to leave behind your honest feedback in a review on Amazon. Feedback is always greatly welcome and reviewed !