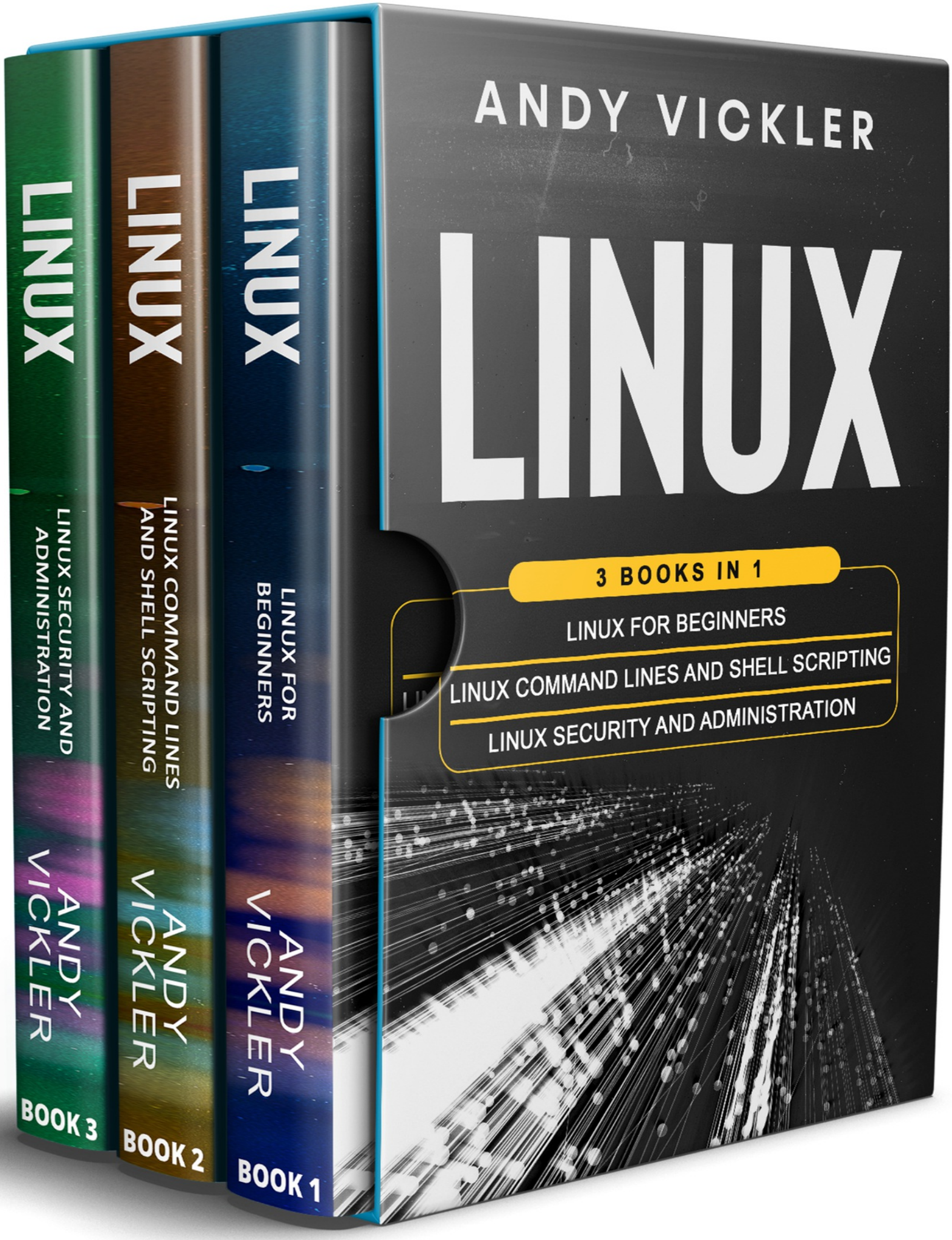
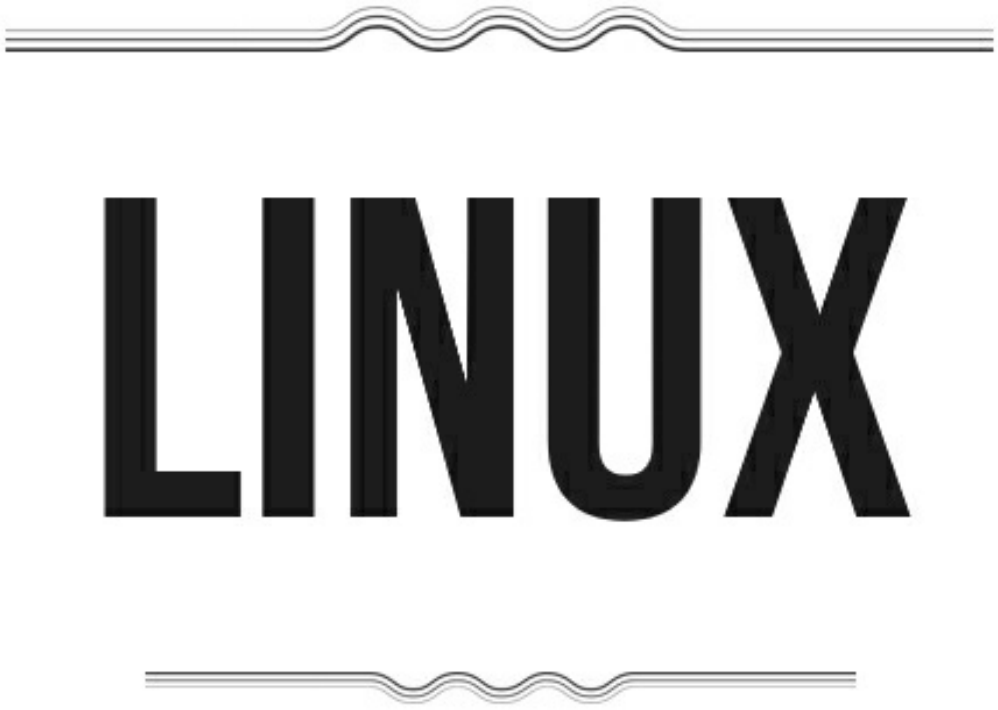

LINUX
LINUX FOR BEGINNERS
ANDY VICKLER
BOOK 1
LINUX
LINUX COMMAND LINES AND SHELL SCRIPTING
ANDY VICKLER
BOOK 2
LINUX
LINUX SECURITY AND ADMINISTRATION
ANDY VICKLER
BOOK 3
ANDY VICKLER
LINUX
3 BOOKS IN 1
LINUX FOR BEGINNERS
LINUX COMMAND LINES AND SHELL SCRIPTING
LINUX SECURITY AND ADMINISTRATION



Andy Vickler

Table of Contents

LINUX
LINUX COMMAND LINES AND SHELL SCRIPTING

LINUX
LINUX SECURITY AND ADMINISTRATION

LINUX
LINUX FOR BEGINNERS

Andy Vickler

Introduction

Linux is an operating system just like Windows, Mac OS, and iOS, but compared to Windows or iOS, Linux is very lightweight and very flexible. Today, Linux can be found everywhere, from smartphones to home appliances, cars to supercomputers, personal computers to enterprise servers. In this book, you will understand everything there is to learn about Linux and why it has become a preferred operating system for everyone, from students to enterprises and businesses.

Linux is the brainchild of Linus Torvalds, a computer science student who wanted to create an operating system that would be free, easy to use, and was capable of serving the needs of every individual in the world. Linux Torvalds was a user of the UNIX operating system and understood quickly that it could be improved to make life easier for everyone. He has proposed his changes to the developers of UNIX, who rejected it. That is when Torvalds decided to develop his own version of the UNIX operating system and call it Linux. He wanted to create an operating system that would take feedback from its users and implanted modifications to make life easy for every user. He collaborated with student programmers at MIT - Massachusetts Institute of Technology and, around 1991, launched a working version of the Linux operating system.

It has been a common misconception that Linux is a very technical operating system, and new users usually shy away from it. But the fact is that in recent years, Linux operating systems have become more user-friendly compared to their counterparts, such as Windows, MAC, etc. There are thousands of Linux distributions available today that can be customized as per a user’s requirements and have even more software on them to make every task easy for a computer user.

This is what Linux is all about, an operating system that makes life easy for you. And this book will help you transition into the world of Linux comfortably; opening a new world of computing possibilities for you as you progress.

Chapter One

Introduction to Linux

History of Linux

In the beginning, a computer would be as big as a car or a house. Naturally, it was a very complicated task to operate a computer. Every computer would have its unique operating system, which made it even more difficult to operate on them. Every operating system would have a specific purpose that would work on a particular computer and could not be used on another computer.

In 1969, a group of developers from Bell Labs decided to develop a common operating system that would work on every computer. They name this operating system 'Unix.' It has a recyclable code known popularly as a 'kernel' today that could be used with any computer. This code was also open-source in nature.

As the years passed, companies like IBM and HP started developing their versions of Unix in the 1980s. The multiple variants of Unix did not gain much popularity, though.

In 1991, a student named Linus Torvalds from the University of Helsinki in Finland envisioned an academic version of Unix and started writing his code. Linux developed a variant of the Unix operating system that is today known as the Linux kernel. This was the project that he started for fun, but little did he know that he was giving the world one of the most powerful tools that would be used forever.

What is Linux?

Linux is a powerful operating system, just like Windows, Mac OS, and iOS. You would be surprised to know that the most popular mobile operating system, Android, is also powered by Linux. An operating system manages the hardware for any computing device such as a laptop, desktop,

smartphone, tablet, etc. In simple words, an operating system handles all the communication between other software and the hardware of a system. Any software would cease to function in the absence of an operating system.

Numerous pieces make up the Linux operating system. Let us go through them one by one.

Bootloader

The bootloader is a module that manages the boot process for a computer. You may have already encountered this without knowing much about it. It’s usually the screen that pops up for a few seconds and then goes away when you switch on your computer.

Kernel

The kernel is the fundamental unit of an operating system that controls your computer's Central Processing Unit (CPU), memory, and other devices.

Init System

This is a section of the Linux operating system that controls the daemons. The most popular init system today is *systemd*. The systemd daemon manages the boot process after the bootloader passes the control over to it.

Daemons

Some background services are initiated during the boot process in Linux. These are called demons and comprise services such as sound, printing, scheduling, etc.

Graphical Server

The Linux subsystem responsible for displaying the user interface on the monitor is known as the graphical server. It is also known as the X server or simply X.

Desktop Environment

The section of the Linux operating system that users interact with is known as the desktop environment. Linux offers a variety of desktop environments, and every environment comes with a set of applications that a user would commonly need to get started with the operating system, such as configuration tools, file managers, games, web browsers, and more.

Applications

A user will not find all the applications they need in the desktop environment by default. Linux has a huge list of applications that can be installed manually. Modern versions of Linux also have something like an app store where you can easily download and install an application.

Why Should You Use Linux?

Almost every new user would have this question in their mind. Why should you use Linux when there are other operating systems like Windows and Mac OS already available? What is the need to learn about a completely new operating system architecture when the default operating system that comes with your laptop or desktop does the job?

We would like to answer this question by putting forward another question to you. Does the default operating system that comes with your computer work fine, or do you face challenges such as viruses, malware, slow speed, eventually? Also, do you get the default operating system for free, or do you have to pay for its licensing? If you struggle with these challenges, Linux is the perfect solution for you. It is a zero-cost operating system for any computer and is very reliable. Yes, it is completely free. You can install Linux on any number of computers without having to shell out a single buck.

If a free operating system is not enough to win you over, how about an operating system that works without any issues forever? For over a decade, I have used Linux for both personal use and as a server system without a single issue of malware, ransomware, or viruses. Linux is not very vulnerable to these attacks. Servers using Linux operating systems need a boot only when the software is updated, and they can go without a reboot for years without causing any performance issues.

Open-Source

We mentioned the word *open-source* earlier. What exactly does open-source mean? Open-source refers to the following aspects.

- You can use the software for anything
- You can study the software and have the liberty to make changes

to it to customize it to your requirements

- You can redistribute the software to any number of people
- You can redistribute the customized version of your software to any number of people

These points will help you understand the Linux community that works together to maintain the Linux platform. In other words, Linux is an operating system that is by the people and for the people. This is also one of the reasons why a majority of users in the world prefer Linux.

Linux Distributions

Given its open-source nature, there are multiple distributions of Linux available today to meet all kinds of user requirements. Linux has ensured that a category of its operating system is present to make everyone happy.

We will be talking about the ten most popular Linux distributions.

Debian

Debian is known as a mother distribution to other popular distributions such as Ubuntu, Deepin, Mint, etc. Debian 10 is the latest major version release for the Debian distribution known as Debian Buster. It is known to provide stability, performance, and an amazing user experience.

The Debian distribution comes packed with more than 59,000 software packages and supports almost all kinds of computer architectures. Users love the Debian distribution for the balance it maintains between technology and stability. There are three branches of the Debian distribution, namely, Stable, Testing, and Unstable.

As the name suggests, the stable version is the one that is made publicly available and is accepted by the users; but it does not come with all the latest applications. It is perfect for users who want a stable operating system and can install the other applications by themselves later. Ideally, Debian Stable is what you would want to go with for your computer.

Debian Testing is a version released at shorter intervals and contains the latest software that is not rolled out to the stable release. The testing version,

as the name suggests, is used for testing the new software so that if vulnerabilities are detected, they can be patched before pushing to the stable release.

The Debian Unstable release is the development grounds of the Debian system. It is experimental and is considered to be a playground for the Debian developers.

Gentoo

Gentoo is a Linux distribution designed for professional users such as developers, system admins, and network admins. It is not an ideal distribution for beginners. It is recommended for users who are already familiar with Linux and want a deeper understanding of the Linux operating system.

Gentoo has a package manager known as *portage*, which is also used by a few other Linux distributions.

Ubuntu

Ubuntu is one of the most popular Linux distributions enjoyed by beginners all over the world. Ubuntu was designed for users who want to transition from Windows or Mac OS to Linux. Ubuntu, by default, has a desktop environment called GNOME that includes every day-use applications such as web browsers, office tools, image editors, media players, and more.

Ubuntu is the foundation for many other Linux distributions such as Lubuntu, Kubuntu, and Linux Mint.

It is the go-to Linux distribution for new users due to its user-friendly interface. Users can readily start using the default applications and start understanding Linux from day one.

Linux Mint

The Linux Mint distribution is based on Ubuntu and is a community-driven distribution. It comes with a very user-friendly and elegant interface and is loved by beginners and professionals. Min is a very stable and powerful Linux distribution.

There are three variants available for the Linux Mint distribution.

1. Cinnamon

2. XFCE
3. MATE

Note that Linux Mint is no longer supported on 32-bit systems and can only be installed on 64-bit systems.

If you want a stable operating system for everyday tasks, you can consider Linux Mint along with your other options. Mint also receives constant updates from the Linux community.

Red Hat Enterprise Linux

Known as RHEL in short, Red hat Enterprise Linux is a distribution that caters to commercial needs. Red Hat Enterprise Linux is a popular choice for server setups due to its stability and regular updates that make it a very secure operating system.

It can be set up on physical and virtual servers and also on the cloud. Red Hat has achieved perfection with containerization technology.

Red Hat also runs a training and certification course, with the two most popular courses being RHCSA (Red Hat Certified System Administrators) and RHCE (Red Hat Certified Engineer).

If you are looking to set up a server with security, efficiency, and stability, you should blindly choose Red Hat Enterprise Linux. RHEL has been using *yum* as its package manager, but with its latest release Red Hat Enterprise Linux 8, it now has *DNF* as its package manager.

CentOS

The CentOS Linux distribution is a community-driven project and provides a reliable and robust operating system. CentOS is based on Red Hat Enterprise Linux and is a perfect alternative since it is free to download and install. It has the same features as Red Hat Enterprise Linux and has free security updates as well.

Fedora

Fedora is a user-friendly and simple distribution of Linux and is very popular with new users. It is a versatile distribution that works with laptops, desktops, servers, and even IoT devices. Just like CentOS, Fedora, too, is based on Red

Hat Enterprise Linux. It is also used as a testing environment for Red Hat Enterprise Linux before the Enterprise version of RHEL is deployed. It is used for research purposes mostly and is a favorite of students and developers.

Fedora also uses the *DNF* package manager.

Kali Linux

Kali Linux is a security-focused distribution of Linux. It is developed by an organization called Offensive Security. The operating system is designed for students aspiring to get into digital forensics and penetration testing. Professionals use it for the same purpose, to review the digital security of enterprises and businesses. Kali Linux comes pre-loaded with all the necessary testing tools such as Nmap, Metasploit, etc.

Kali Linux is the Linux operating system for cybersecurity professionals or students who want to make a career in cybersecurity. Kali Linux also runs certification courses such as Penetration Testing with Kali and Kali Linux Certified Professional.

Kali Linux uses the *APT* package manager.

Arch Linux

The Arch Linux distribution is a geeky lightweight distribution designed for advanced to expert users who are not bothered with the default software and service running on the operating system. It gives the user a whole lot of options to play with the configurations and customize the operating system as per their preferences. Arch was designed for veteran Linux users who know the ins and outs of the operating system.

Arch Linux follows a rolling release and receives regular updates. This means you can install any version and update it immediately to the latest version. The package manager for Arch Linux is known as *Pacman*.

OpenSUSE

The OpenSUSE is a modern distribution developed and maintained by a very comprehensive Linux community. It has two branches, SUSE Leap and SUSE Tumbleweed.

SUSE Leap has been designed for desktop users. It is also a good choice for

testing and enterprise development purposes. It is popular with open-source developers and system administrators.

SUSE Tumbleweed is a rolling release that includes all the latest software stacks and Integrate Development Environments (IDEs) and is a brilliant distribution. It is a very good option for a developer or a power user, given that it has all the up-to-date development tools.

The package manager in OpenSUSE is known as *Yast*.

These are just the most common and popular Linux distributions, and there is no exhaustive list as such. There are more than 600 distributions in the market for Linux and over 500 that are in development.

Linux vs. Windows vs. Mac

A business invests a lot of money to ensure that its systems are secure. Cybersecurity solutions include dedicated security applications for firewalls, identity management, etc. When it comes to operating systems, anti-malware and antivirus software is installed on every system, but a lot of how secure a system is depends upon the features of the operating system itself. Is a Windows system more secure, or is a Mac system more secure? In this section, we will compare the major three operating systems - Windows, Linux, and Mac - from a cybersecurity standpoint.

Windows

According to us, Windows is not considered less secure than Mac or Linux because it has low-security standards or Microsoft does not put enough effort into its security, but because it is used on a large scale in organizations. The number of Windows users is massive due to which attackers target the Windows operating systems the most. Given this fact, there is new malware developed to attack Windows systems every day. Technically speaking, Windows is as secure as other operating systems. Moreover, the security team for Microsoft deploys security patches for Windows almost every week. Believe it or not, Windows has the biggest database of malware and virus signatures. Despite this, attackers have been persistent in exploiting any unpatched vulnerabilities that are discovered on the system for even a short duration of time. This being said, the Windows operating system does not come out of the box with any issues that you may choose other operating

systems over it. It’s just that attackers target Windows systems over Linux or macOS because they stand a higher chance at success on Windows systems.

In the past few years, Microsoft has become very proactive in developing and rolling out patches to Windows systems regularly. Windows comes with preloaded antimalware software capable of detecting all kinds of malware and virus signatures.

Windows also maintains a sandbox environment for its store that safeguards a system from malware and viruses that other security solutions may miss. Windows also uses a hashing mechanism to check if any data on the system has been tampered with. The hashing process happens when an application is installed and run for the first time.

macOS

macOS is known for being a secure operating system by default. But this only implies that it does not have multiple network services running right after installation that can be exploited by attackers. Apple has integrated the T2 Security Chip with all its latest devices that make the macOS safer than the older devices. The Apple T2 chip has the Secure Enclave coprocessor, which implements the FileVault security, secure boot, Touch ID, and storage encryption. The T2 chip also does not let free or open-source software launch by default on the macOS. The security implementations on macOS revolve majorly around the boot process, the operating system's real-time operations, and software updates.

The macOS systems also face fewer problems concerning viruses than Windows systems, but this does not mean that MacOS systems are completely secure from malware. Vulnerabilities are encountered in MacOS from time to time as well. The popularity of Windows systems connecting to the Internet is far more compared to Mac or Linux systems. As a result, naturally, the number of attacks on Windows systems is more, but in recent years, MacOS has been gaining market share, and attackers have started noticing the Apple operating system as well.

Apple also has another security feature called System Integrity Protection (SIP), introduced after its 2015 operating system release. It includes security implementations that are enforced directly by the kernel. This process protects against changes that a process can force into the system.

Linux

Unlike Windows and macOS, the Linux operating system is completely open-source. This means that millions of people in the world are playing with the Linux code every day. The Linux community is constantly looking for any vulnerability in the operating system and patches them as soon as possible. Obviously, the more people who review the code, the more secure the operating system becomes. In contrast, if you have a small team reviewing code for operating systems, as in the case of Windows and macOS, it is natural to have issues, and the number of vulnerabilities will also be more.

It is a popular belief that Linux is far more secure compared to Windows or macOS, and this is because Linux offers multiple options to run any process in a sandbox environment before it can be run in a live environment. Linux also has numerous security aspects that run hand in hand with each other. Before even implementing security using firewalls, Linux solves 99% of security issues by just implementing permissions.

For instance, let us talk about Fedora, which is a popular distribution of Linux. Fedora has a default feature called Security-Enhanced Linux, also known as SELinux in short. SELinux applies security policies, access controls, and many more security features automatically. Fedora also has a compiler feature called Position-Independent Executable (PIE) that creates a hardening wrapper around all its processes. This process is also known as security hardening.

Contrary to popular belief, vulnerabilities in Linux can be discovered and patched instantly because of its open-source nature, but Linux is missing additional security measures such as sandboxing and hashing. Most people in the world fail to trust the Linux operating system because it is open-source, free, and has limited security support. Many businesses believe that since people can play with open-source code, it is not secure, but this logic is incorrect. Most web servers in the world today run on the Linux operating system specifically because it is much more secure against attacks.

What can we take away from this comparison?

More than 75% of desktops run versions of the Windows operating system worldwide, with Apple having around a 10% market share. Concerning the

code and functions, Windows and macOS are very different operating systems. The latest versions of Windows use the Windows NT kernel, whereas macOS uses the UNIX kernel.

If we are talking about vulnerabilities in Windows, Linux, or Mac, or any other operating system, they are all very similar. Developing an operating system is a very complex task, and all of them will encounter similar security issues. You can say that a Mac system is not necessarily more secure than a Windows system. The point to focus on here is what the attacker's target is. Suppose the attackers want to target as many people as possible worldwide. In that case, they will not be interested in operating systems such as Linux or macOS, which have a smaller user base.

macOS does not have anything, in particular, that would make it less secure. What differentiates Windows, Linux, and macOS is that malware be loaded on any of these must be coded in a separate format before being installed. In this case, one size does not fit all.

It's very simple. Attackers target operating systems that have the biggest user base. Naturally, most malware present in the world today is developed for Windows systems. This means that if a Mac or a Linux user receives malware in an email meant for Windows users and clicks on it, it would not even launch since the malware code is not recognized by macOS or Linux. This doesn't mean that there is no malware out there for macOS or Linux systems, but it is very rare. The bottom line is that MacOS and Linux systems are more secure than Windows systems, but not for the reasons everyone in the world thinks they are.

Also, attackers shy away from Linux systems because it has a very low user base. It has less than 5% of the market share in the operating systems market. Linux also does not give admin access to its users by default, thereby protecting damage that users can do by being naive. Also, Linux has a huge community working on discovering vulnerabilities every day, resulting in quickly patching the operating system.

Every operating system has pros and cons. It depends on an individual or an organization to pick up the operating system that best suits their requirements.

Chapter Two

Ubuntu

We have discussed the various distributions of Linux in the previous chapter. We have learned that Ubuntu is one of the best Linux distributions for new users to transition smoothly from Windows to Linux. We will be using the Ubuntu Linux distribution throughout this book.

Let us quickly go through the history of Ubuntu.

In April 2004, around a dozen developers from GNOME, Debian, and GNU Arch projects came together to brainstorm under Mark Shuttleworth's leadership. Shuttleworth only had one question for them – whether it was possible to create a better operating system. And all of them had the same answer, “Yes.” They then brainstormed over how it would look like and talked about the community responsible for this operating system. The group worked to define the answers for these questions put forward by Shuttleworth and decided to turn these answers into reality. They named their group the Warthogs and gave themselves a 6-month timeline to develop a proof-of-concept operating system. The first release of their operating system was named the Warty Warthog as they assumed that the first release would have its warts. After this release, they got down to real business.

If you get in touch with anyone privileged enough to see the first release of the Warthogs, they’d tell you about how fulfilling it has been to see the progress made by this operating system over the years. Ubuntu had a very strong beginning and had surpassed everyone’s expectations. Within no time, Ubuntu made it to the number one ranking of GNU/Linux distributions. Ubuntu had amazing growth compared to any GNU/Linux distribution and experienced an impressive first year. Even after that, it continued to grow over the years.

It is surprising to see that just after a few years of its launch, millions of users had already moved to Ubuntu. And thousands of users from these millions

also give back to Ubuntu by developing code, documentation, and translations. Given this, Ubuntu is improving every day.

What Does Ubuntu Mean?

The Warthogs had a good team and a vision for their goals, but the team did not have a name for the project. It was Shuttleworth who insisted that the name be Ubuntu.

Ubuntu is a term and a concept that comes from numerous South African languages such as Xhosa and Zulu. It refers to the South African ethic or ideology that can be translated to “humanity toward others" or “I am who I am because of who we all are.” A few others have translated Ubuntu as “the belief in a universal bond of sharing that connects all humanity.” Archbishop Desmond Tutu, a human rights activist from South Africa, described Ubuntu as follows.

> *“A person with Ubuntu is open and available to others, affirming of others, does not feel threatened that others are able and good. He or she has a proper self-assurance that comes from knowing that he or she belongs in a greater whole and is diminished when others are humiliated or diminished when others are tortured or oppressed.”*

Shuttleworth had numerous reasons for choosing the name Ubuntu for the project.

1. It was a South African concept. The people working on Ubuntu may not be from South Africa, but the project's roots are. Shuttleworth wanted a name that represented this.

2. The project is about relationships with others and lays a foundation for a community that shares and collaborates. These are the core pillars of free software.

3. The Ubuntu team wanted to build a highly functional community that would have personal relationships built on connections and mutual respect. The term Ubuntu would tell people how the project was formed and the future vision for it.

Due to all these reasons, the name was perfect, and it stuck for good.

The Vision

The full-time developers of Ubuntu had to be paid, and Shuttleworth needed a company through which he could employ them. Shuttleworth wanted the best developers from the free and open-source community to work toward the development of Ubuntu. The developers would be from all over the globe, resulting in the elimination of geographic or national boundaries. Rather than having a physical company where everyone would sit together, Shuttleworth decided to employ the developers through a virtual company. Although this had its cons, like different time zones, bandwidth and latency issues, etc., it also had many benefits. The employees' distributed nature meant that the company could recruit new employees from anywhere in the world without asking them to pack their lives and move to a new country for work. This also eliminated the common water cooler problem, where employees in an office usually spend time at a water cooler having personal conversations. Since this was a virtual company, employees would have personal conversations over messengers while simultaneously working on the project. The company was named Canonical, and the closest thing to an office it had in the initial years was Shuttleworth's residence in London. Today, the company has multiple offices around the globe, but most of its employees work from home. Needless to say, the developers depend a lot on Internet collaboration.

The Ubuntu Community

The credit for what Ubuntu is today goes to the Ubuntu community. We already know that the definition of Ubuntu also revolves around people and a community. We will be discussing in a dedicated chapter how you can become a part of the Ubuntu community.

Goals and Promises

Philosophical Goals

Ubuntu's most important goals are philosophical. You will find the philosophy of Ubuntu documented in detail on their website. We have put down the main philosophies from their website verbatim below.

Ubuntu's Philosophy

Ubuntu is a philosophy-driven Linux operating system, and its aim is to distribute free software across the world so that humans can benefit from it. The core ideas of the Ubuntu philosophy are as follows.

1. A user should be able to download, install, distribute, use, and modify software for any requirement, and they should be able to do this for free without having to pay any licensing fees.
2. Every user who owns a computer should be able to use any software on it in a language they prefer.
3. Even if a user has a disability, they have the right to use a computer with software that supports their disability.

These core ideas of Ubuntu's philosophy are reflected in their operating system. When you install Ubuntu on your computer, the base operating system and all the software that comes with it already meets the above-mentioned ideals. Ubuntu constantly strives to provide software to users with a license type that gives the users complete freedom.

Free Software

In Ubuntu, the word free in free software refers to the freedom to use software and not about costs, although Ubuntu is still committed to providing software without any charge. Ubuntu's main objective with free software is to give complete freedom to its users who use their software. This freedom empowers Ubuntu users to grow and have a beautiful experience.

What is free software? According to the Free Software Foundation, free software has the following set of freedoms.

- The freedom to run any software for any requirement
- The freedom to understand the software and make it meet your needs
- The freedom to redistribute the software to help everyone else
- The freedom to modify the software and launch it publicly such that everyone benefits from it

Open-Source

The term Open-source was introduced in 1998 to differentiate it from the English word ‘free.’ The Open-source Initiative defined open-source software. Ever since, open-source has become very popular and continues to grow.

Ubuntu is an open-source operating system. There have been arguments if the free and open-source are the same thing, but Ubuntu maintains that it does not see free and open-source as incompatible or different. Ubuntu supports individuals from both movements.

If you read through the goals above, you will realize that Ubuntu makes it clear that a user should have the freedom to use free software. Why is this so important? Firstly, a user benefits practically from faster, better, and more flexible software than other operating systems. A user also transcends their role as a consumer and user of the software. Ubuntu wants to empower software and make it work as required by its users. The software has to be free, and Ubuntu includes this in its philosophical goal.

Free software is not the end of the goals proposed by Ubuntu. It also has two other equally important goals. The first one states that every computer user should be able to use a computer in a language that suits them best. This is a nod to the fact that not everyone in the world knows English, but still, the majority of software in the world is only in the English language. There is a lot of textual content in software, and it has to be written using a language; but if this is only in English, many people in the world will not be able to use the software at all. A computer is a tool that educates and empowers, but it can do so only if the user understands its interface. Ubuntu believes that it is the people and the Ubuntu community's responsibility to ensure that any user in the world can use their language of choice when they use Ubuntu. This translation is made possible through the first philosophical point that Ubuntu makes about the ability to make modifications, which is a feature of free and open-source software.

Finally, just as a user should not be restricted from using software because of language constraints, they should also not be restricted from using a computer because of any disability. Ubuntu ensures that it is accessible to users with vision disabilities, motor disabilities, or hearing disabilities. All of

these disabilities are taken into consideration and alternate methods of input and output have been developed. There are a significant number of intelligent people in the world who have some kind of disability. Ubuntu welcomes individuals with disabilities to be a part of the Ubuntu community to contribute to it and make Ubuntu an operating system that is truly meant for everyone.

Conduct Goals

If Ubuntu's philosophical goals describe the why of the Ubuntu project, it's *how* would be described by the Code of Conduct. The code of conduct for Ubuntu is a very important document that governs the policies and cooperation in the Ubuntu community. You can become an Ubuntu activist if you are in complete agreement with this document. It is also an important step to become a member of the Ubuntu project.

The Ubuntu code of conduct covers "behavior as a member of the Ubuntu community, in any forum, mailing list, wiki, Web site, IRC channel, install-fest, public meeting, or private correspondence." The Ubuntu code of conduct also goes deeper into some points that bear the following headings.

- Be considerate
- Be respectful
- Be collaborative
- When you are unsure, ask for help
- When you disagree, consult others
- Step down considerately

You may also say that all these headings are mostly common courtesy or common sense. Nothing in Ubuntu's code of conduct is radical or controversial. These heading are there intentionally, but it can be difficult for everyone to follow this. Ubuntu does not believe in enforcing this code of conduct since acting respectfully, collaboratively, and considerately is a personal choice. Ubuntu's code of conduct is not designed to be a law that prohibits any language or actions. It is in place to just serve as a reminder that

respect and collaboration goes a long way in developing and maintaining a healthy project and an even healthier community.

Nobody in the Ubuntu community is above the code of conduct, not even Mark Shuttleworth. The code of conduct cannot be waived and is not optional. Ubuntu also has the Leadership Code of Conduct that extends a few more expectations to those in the leadership positions in the Ubuntu community, but neither the code of conduct nor the leadership code of conduct was designed to eradicate disagreement and conflict. Arguments in the Ubuntu community are as common as any other online community. The one thing that differentiates the Ubuntu community from other communities is that it ensures that the arguments happen in a healthy mutual respect and collaboration environment. This leads to better arguments and better results, ensuring that feelings are less hurt and egos are less bruised.

There have been instances where the code of conduct and leadership code of conduct have been used incorrectly, but they cannot be used as sticks while arguing with an opponent. They should be used as pointers to gain the consensus of everyone. Code of conduct violations is rare. When the group feels that a group member is not aligned with the code of conduct, they will give a gentle reminder to the member, privately, that the code of conduct is in effect. This is enough to put an end to a conflict almost every time. Code of conduct violations are rarely brought to the Community Council.

Technical Goals

Although we have seen how a respectful community and compliance to the code of conduct form the Ubuntu project's backbone, we have to understand that Ubuntu is a technical product at the end of the day. It would be sensible for Ubuntu to have some technical goals along with philosophical goals.

The first technical goal for Ubuntu and the most important one is to ensure releases at regular intervals. As we saw before, in April 2004, the Warthogs team had set a release date within six months of their initial meeting for the proof of concept release. Since then, the group has always stuck to the six months release cycle with the only exception for the LTS release, where the release was extended by an additional six weeks to ensure that everything was right. The extension was also approved only after seeking approval from the community. Also, since an additional six weeks were taken on this

release, the team released the next version in a mere four and a half months. It is important to have frequent releases so that users can keep enjoying the latest free software. Predictable releases help businesses using Ubuntu to plan changes for their business. Regular releases help businesses using Ubuntu, a reliable software through which they can grow and expand without worrying.

While regular releases are important, it is also important to support software after its release. Every software has bugs, and Ubuntu is no exception. Bugs in any software after a major release are minor, but fixing them can also introduce new bugs that are much worse. After any software is released, bugs should be patched with care or not be fixed at all. The Ubuntu community works on fixing between major releases only if the changes can be tested thoroughly. Most of the time, bugs can lead to loss of user data or affect the operating system's security. These bugs are fixed instantly by the Ubuntu community and are pushed to the systems via updates. The Ubuntu community also ensures that the bugs are minimal when a version is released and also is terrific at fixing them if any are found, but there is always a possibility that new bugs can come into the picture. The Ubuntu community readily commits to supporting every major release for 18 months from the time of release. Also, in the case of LTS releases such as the original Ubuntu 6.06 LTS release, the community went beyond the usual 18 months and supported the operating system for three years for desktop systems and five years for server systems. This became so popular with individual users and businesses that the three years and five years pattern is continued to date for Ubuntu’s LTS releases.

The Ubuntu project's third major technical goal is support for both desktop and server systems in separate but equal modes. Ubuntu is popular in desktop systems, but a group of Ubuntu developers works on improving Ubuntu for both desktops and servers. The Ubuntu community provides installation media for both these systems as they consider both to be essential. The software for both types of systems is tested extensively, and documentation is provided as well. In this book, we will go through using Ubuntu for both desktop and server systems.

Finally, the Ubuntu project's last technical goal is to make it easy for users to transcend their roles as users and consumers of free software by taking advantage of the freedom that is part of Ubuntu’s philosophy. Because of

this, the development of Ubuntu has always revolved around using one programming language - Python. Ubuntu developers have ensured that Python is extensively used throughout the operating system. Ubuntu developers have used the Python programming language in the desktop applications, text editors, console applications, and the overall system of Ubuntu. This has made it convenient for users to learn only one single language to take advantage of the system, automate tasks, and make modifications as per their requirements.

Beyond the Vision

An introduction to Ubuntu is incomplete without a brief discussion of its derivatives. The Ubuntu flavor of Linux is based on Debian, but Ubuntu has its sub-flavors as well. They are as follows.

Kubuntu

Kubuntu is a sub-flavor of Ubuntu and is an alternative for Windows or macOS that contains all the necessary applications to enable a user to work, share, and even play. Unlike Ubuntu with the GNOME desktop environment, Kubuntu uses the KDE desktop environment, and its installation comes packed with applications for email, office, photography, music, etc.

Edubuntu

Edubuntu is another sub-flavor of Ubuntu that has been developed to encourage computing through educational software in schools. There have been various changes in the Edubuntu operating system every year, but the prime focus of the Edubuntu project remains the same: to provide free and educational software to schools. Edubuntu has a feature through which you can have a single operating system running on a server computer in a classroom, and every other student can connect to it through a client system and have their own session on the server computer. This has helped schools offer computing systems to their students while keeping the costs for having such a setup to a minimum.

Xubuntu

Xubuntu is another derivate of Ubuntu and is maintained by a dedicated community. It was designed to have an elegant and user-friendly interface. Xubuntu is aimed at users who want to have a good-looking desktop and also

want to perform their daily tasks with ease. The best part about Xubuntu is that it is so lightweight that it works on computers with older hardware too.

We have discussed how one person and a team's vision to create a beautiful operating system with a philosophy became a phenomenon. We understood how the vision for Ubuntu was not merely to be a technical product but to unite the human community via means of an operating system that encouraged mutual respect between the users and the community that built it.

Chapter Three

Installing Ubuntu

Trying Ubuntu is very easy. You can try Ubuntu directly from a media like a USB stick without even having to install it on your computer's hard disk. This is a good option if you already have another operating system installed on your computers, such as Windows or macOS. You can simply run Ubuntu from a USB stick and don't have to worry about the installation overwriting your existing operating system.

Choosing an Ubuntu Version

The Developers of Ubuntu have ensured that the operating system they built is easy to install. They had already considered that Ubuntu users would spread across a wide spectrum with all kinds of users with multiple purposes who would want to install Ubuntu on various systems. There are various versions of Ubuntu that are available on the official Ubuntu website. For this book, we will be taking into account the installation of Ubuntu 20.04 LTS that was released in April 2020. Given that it is an LTS release, it will receive support from the Ubuntu community up to April 2025. This version of Ubuntu supports the three major types of computers with a few more variations.

i386

This is the architecture of Intel processors and includes Apple hardware as well. If you are uncertain of which version to download, select this one. It will work on both 32-bit and 64-bit Intel processors.

AMD64

This is the architecture of AMD processors, and if you have an AMD processor on your computer, this is the version you should choose, as it will suit your hardware efficiently.

ARM

ARM refers to low powered processing chips that are found in smaller devices such as a Raspberry Pi, tablets, and mobile phones. There is an agreement between ARM and Ubuntu to keep developing Ubuntu for ARM chips, and that has made Ubuntu the first major distribution that supports ARM devices.

Feature of Ubuntu 20.04 LTS

Linux Kernel 5.4

With the Linux kernel 5.4, support for a wider range of processors has been added to Ubuntu 20.04 LTS, and power-saving, boot speed, etc., have been improved as well. Support for USB-C and several other security features have been added to this release as well.

GNOME 3.36

The desktop environment for Ubuntu, GNOME, has been updated and improved as well. System animations are now smoother while putting smaller loads on the CPU.

ZFS 0.8.3 file system

The file system's performance has been improved, and encryption is now a default feature.

New Login Screen

The login screen has been redesigned

Updated Versions of Programming Languages

Python 3.8, PHP 7.4, OpenJDK 11, Rustc 1.41, Glibc 2.31, Ruby 2.7.0, GCC 9.3, Golang 1.13, Perl 5.30 have been added to this release.

Prerequisites to install Ubuntu 20.04 LTS

The following system requirements will be needed. These are recommended resources.

- A 2 GHz dual-core processor
- 25 GB available disk space
- 4 GB memory

- USB support

A USB stick of a minimum of 4 GB is also needed to create the Ubuntu installation media.

Installation

Let us go through the steps of downloading and installing Ubuntu on a computer.

Downloading the Installation Media

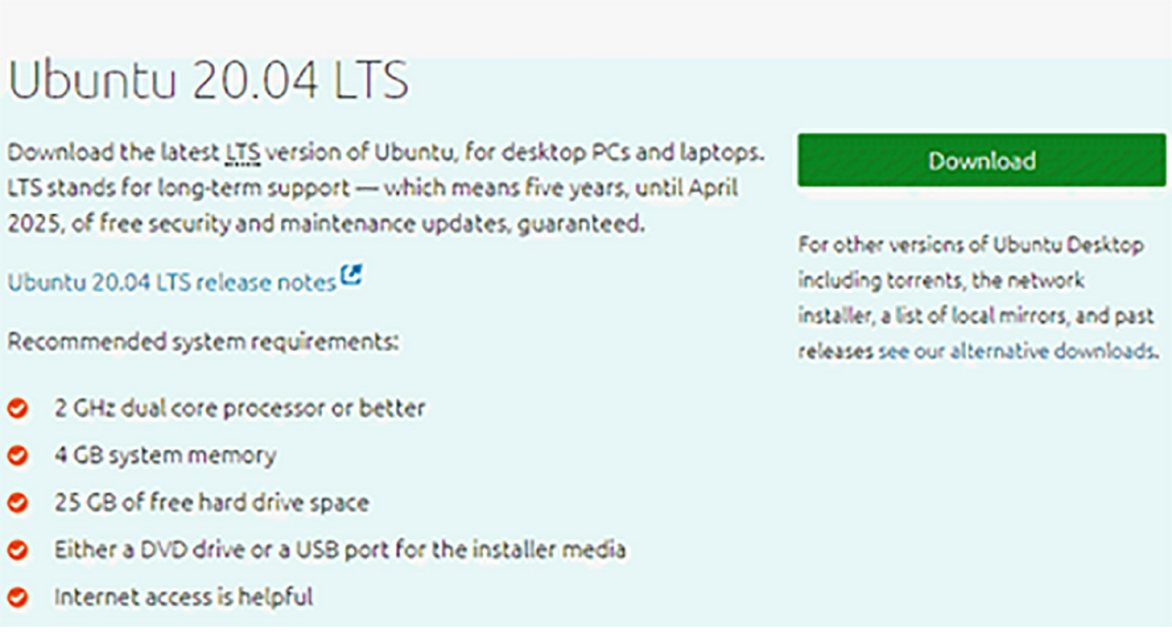
Open a web browser and navigate to https://ubuntu.com/download and select a version that suits your hardware, as we have discussed earlier. The most popular versions of Ubuntu are:

Ubuntu Desktop

Ubuntu Server

Ubuntu Derivatives

After locating the version for your system, click on the green download button next to it. You will be redirected to a thank you page, and the download will begin. We will be downloading and installing the Ubuntu Desktop version.



The downloaded file will be in a .ISO format. We will be using this file to create a bootable USB drive.

Save the file as per the location you need.

Creating a Bootable USB Drive

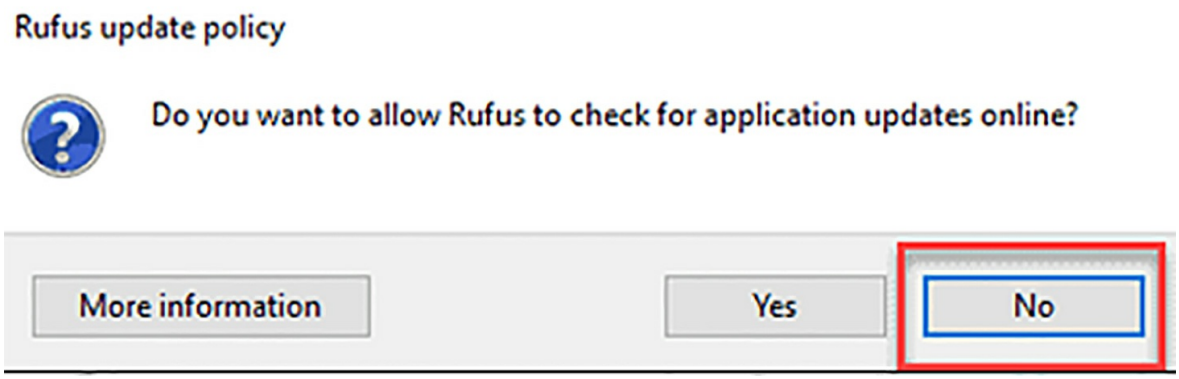
As mentioned earlier, you will need a USB drive with 4GB or more space. This process will also delete all existing files on your USB drive. Make sure you have backed up any important data that is present on your USB drive. We will also be creating this bootable USB on a Windows system.

You will need a third-party application on Windows, such as Rufus, to create your bootable USB drive.

You can download Rufus from https://rufus.ie/ Scroll down to the section for downloads and click on the latest version of Rufus to begin downloading.

Launch the file after the download is complete.

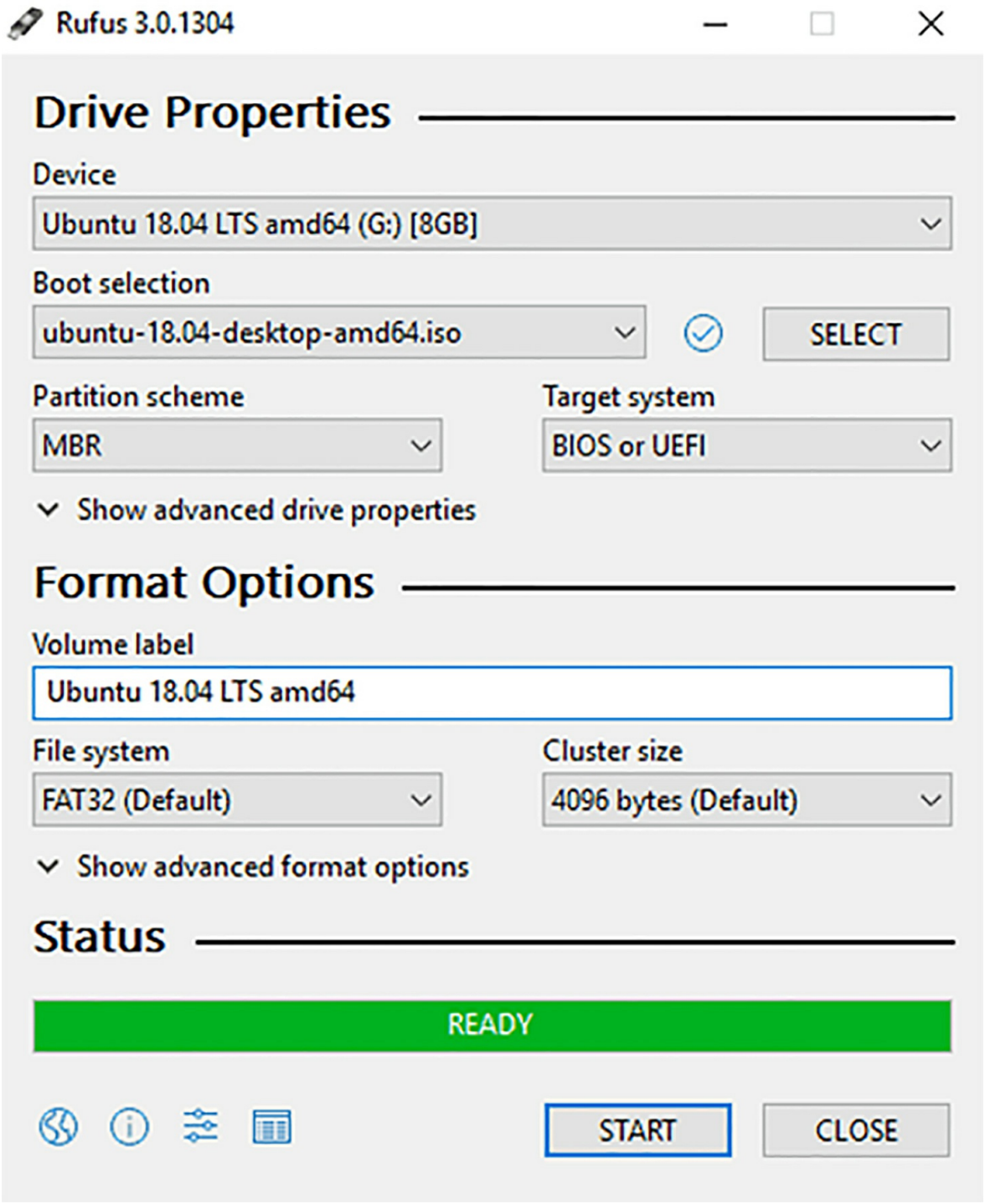
You will get a pop-up. The pop-up will ask you if you want to check for any online updates. Click on No.



Once the Rufus application has launched, plug in your USB drive to your computer. The application will recognize it, and it will be listed in the Device field.

Select the USB drive in the Device field

In the dropdown for Boot Selection, select Disk or ISO image. Click on the Select button next to it. From your computer, select the Ubuntu ISO that you downloaded earlier.



Click on the Start button.

This process will write the Ubuntu installation files to your USB drive, and it will become a bootable USB drive as well.

Booting Ubuntu from your USB Drive

1. Switch off your computer and remove all other USB devices attached to it.
2. Insert the newly created bootable USB drive and switch on your computer.
3. Now either your computer will automatically boot from the USB drive, or if it doesn't, you will need to go to the BIOS settings for your computer and configure boot devices so that the computer can boot from the USB drive.
4. Every computer manufacturer has different ways to get into the BIOS settings when the computer boots up. You should be able to see the keyboard key to be used to go to the BIOS settings for a couple of seconds when you switch on your computer.
5. If everything goes as intended, you will see a boot prompt that will ask where you want to boot from. Select the USB drive. You should now be able to see the Ubuntu live disc menu.

Run Ubuntu from the USB Drive

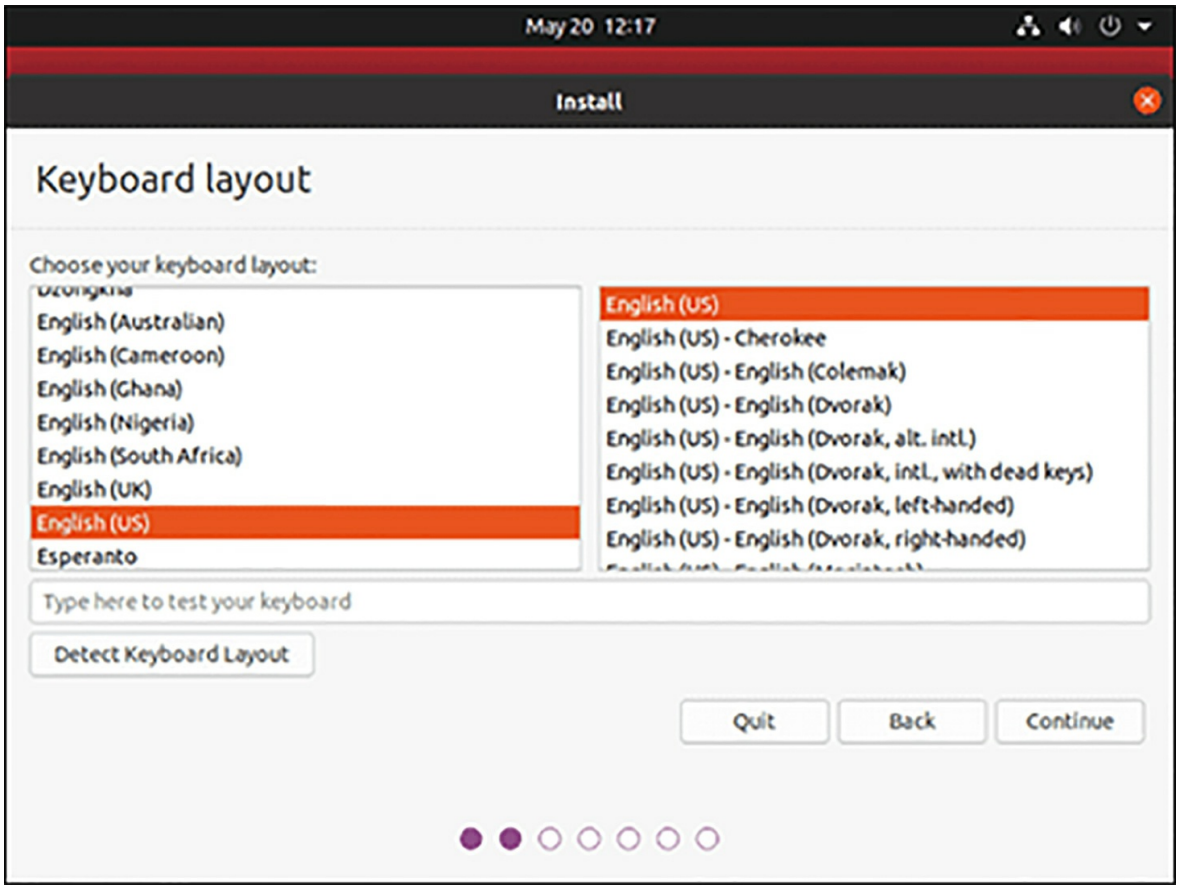
You can run Ubuntu directly from the USB drive before you choose to install it to your hard disk. The .ISO has a live mode that will run directly from the USB drive. You can launch this mode by clicking on Try Ubuntu.

Install Ubuntu 20.04 LTS Desktop

You can begin the Ubuntu installation by clicking on Install Ubuntu

Select the Keyboard Layout

The default selection on this screen will be English and English. You can change the layout if you are used to a different keyboard. Click on Continue when you are ready.



Applications to Start With

Normal Installation

This will have the complete Ubuntu Desktop installation that includes all software, media players, and games.

Minimal Installation

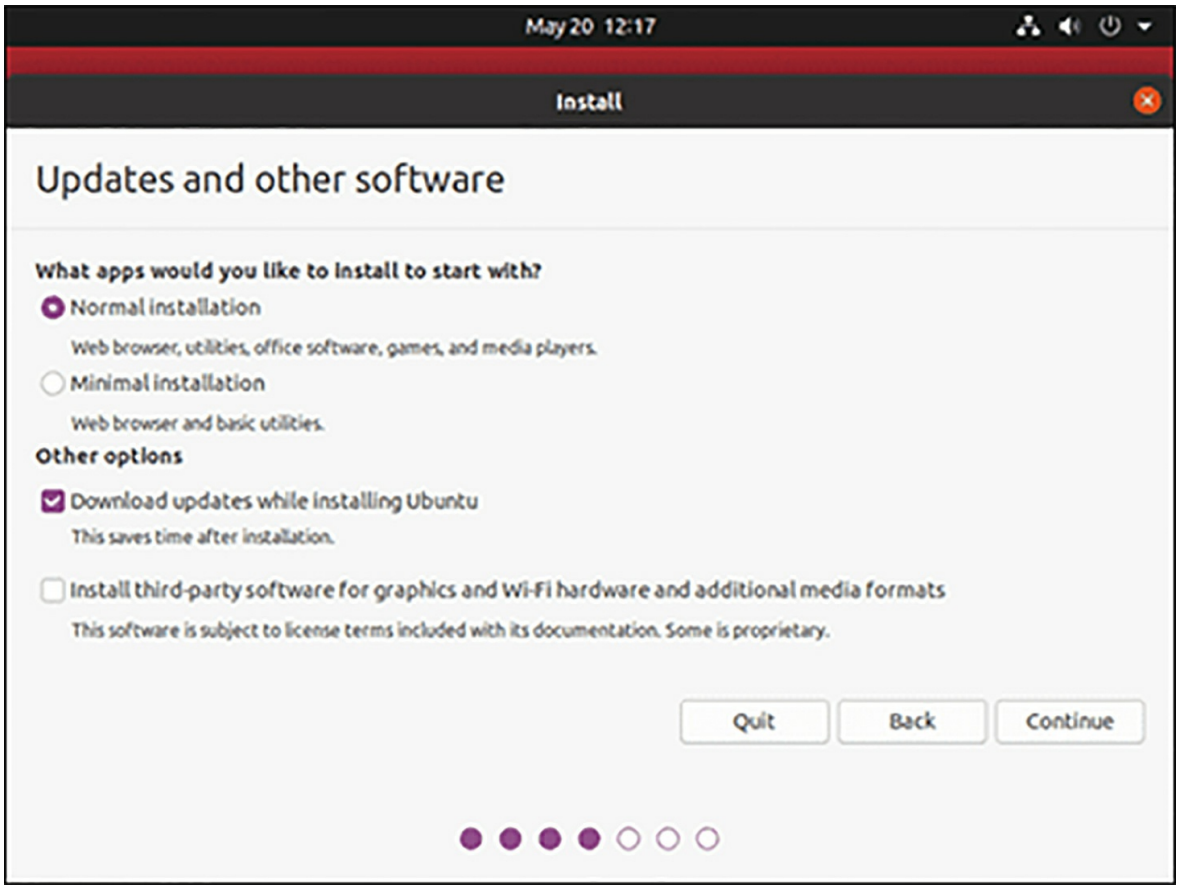
This option will install the minimal version of Ubuntu Desktop with limited software. You will also need to confirm some more options as follows.

Download Updates while Installing Ubuntu

Sometimes the installer needs to download additional files from the Ubuntu repository. They are not mandatory, but this option saves you time, and the system gets updated right at the time of installation.

Install Third-Party Software for Graphics and Wi-Fi Hardware and Additional Media Formats

The system you are installing Ubuntu on may have additional hardware such as a graphic card and a Wi-Fi network card. The open-source drivers included in the Ubuntu installation may not support your hardware by default. This is when this option should be enabled so that the installation downloads third-party drivers for your hardware if required.



Disk Partitioning

The next screen gives you a window to choose the type of installation. The first option lets you do an Ubuntu installation from scratch. Note that this will erase everything on your hard disk and install Ubuntu. If this is what you wish to do, you can stop reading here and go to the next step.

If you already have sufficient knowledge about operating system installations, you may also want to look at the Advanced Features. You can use this option if you want to create custom disk partitions and for some additional options such as:

Use LVM with the New Ubuntu Installation

LVM or Logical Volume Management is a tool that can be used to create virtual drives on Linux. It is very similar to the gparted tool.

Encrypt the New Ubuntu Installation for Security

You can enable this option if you wish to encrypt all the data on your hard drive, including the installation files. You will get an option to create a decryption key that can be used later to decrypt the data.

If you wish to create custom partitions, you can select Something Else.

When you select this option, you will be taken to a screen to create custom partitions. You can create physical partitions, and Linux will consider these partitions as individual physical hard drives.

You can apply the changes you made to the disk partitions by clicking on Continue.

You will receive a confirmation box that says Write changes to disks? This is your last chance to review the partition changes, and only after you click Continue on this screen, your partitions come into effect.

Selecting the Time Zone

When the installation concludes, you need to set up your timezone.

Type your city's name in the box. This will help the system set the time zone for your system. Click on Continue.

You will be able to change the timezone later as well from the Ubuntu desktop.

Create User Account

On the next screen, you will be prompted to create a user account. You will need to enter the following details to create a user account.

Name: Type in your name here

Computer Name: You can provide a hostname for your system here

Username: This will be the account name for your user

Password: Enter a password of your choice and ensure that it is

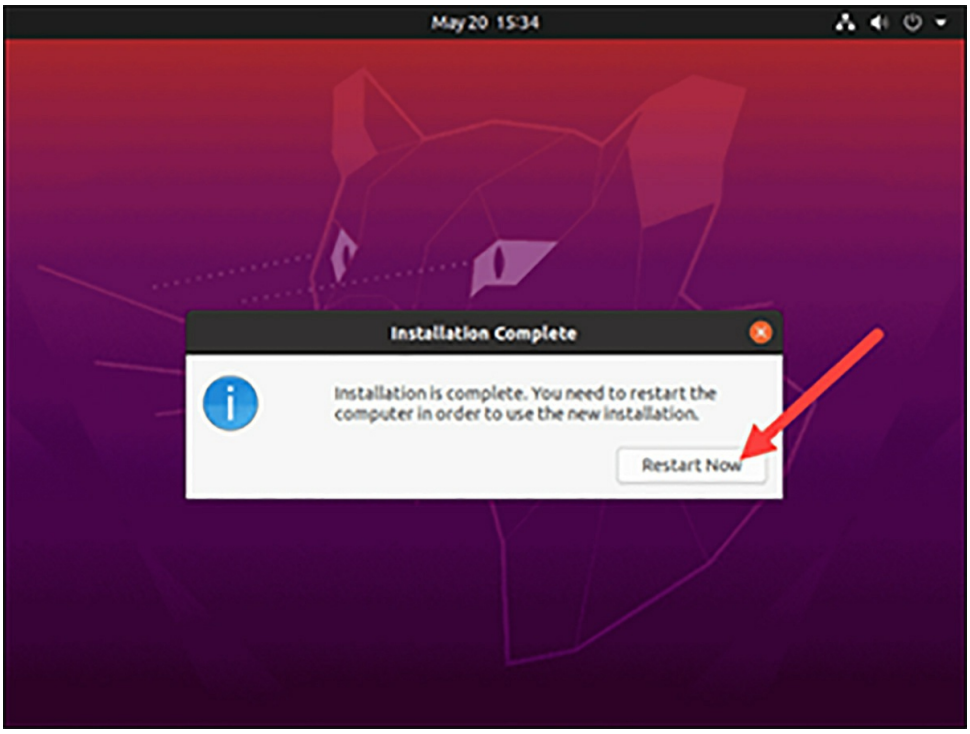
strong. The installer will let you know the strength of your password.

Log in Automatically: This option should not be enabled ideally.

Require my password to log in: Ideally, this option should be enabled.

Click on Continue to process with the installation.

The installer will start writing Ubuntu files to your hard disk, and this can take some time depending upon your system's hardware. After the installation is complete, unplug the bootable USB drive from your system. You will be prompted to restart the system. Click on Restart Now.



The system will restart and should now boot into the newly installed Ubuntu 20.04 LTS.

You have now successfully installed Ubuntu on your system. If your system is connected to a network via Ethernet or Wi-Fi, you will see prompts to download and install updates. It is advisable to install the updates and keep your system up to date.

In this chapter, we went through the most common Ubuntu installation

wherein a bootable USB drive is created, and Ubuntu is installed from scratch on your system, which may have had Windows before. There are other methods to install it, too, wherein you can install Ubuntu and still keep your Windows operating system as well. This is done by creating the Ubuntu installation on a different partition than the Windows operating system installation. We do not cover that kind of installation in this book, as this is a book for beginners.

Chapter Four

Getting Started with Ubuntu

Now that you have installed Ubuntu, it is time to start using your Linux operating system. Unlike the paid operating systems such as Windows or macOS that require you to pay more to get additional software, Ubuntu already comes loaded with the software you need to get started. You will find everything in Ubuntu right from an office suite, a web browser, to email media tools. You are good to start with using Ubuntu as soon as the installation is completed.

Everyone uses a computer in their unique way, and every user likes to customize the look and feel of their computer to their requirements. Keeping this in mind, Linux lets you use one of the many graphical interfaces that it has to offer. Thus, there are hundreds of graphical interfaces available on Linux to suit every user's needs.

Although there are multiple graphical interfaces available for Linux, two of them are the most popular ones, GNOME and KDE. Each of these environments has a user-friendly and easy-to-use interface. They have a few differences concerning the look of the desktop and how they can be personalized further.

The KDE environment gives a user complete control and configuration options to personalize the desktop. The user can configure every aspect of the desktop and has the liberty to modify the desktop's look and feel.

GNOME, on the other hand, is inspired by the desktops of Windows and macOS and prioritizes simplicity. GNOME can be customized easily, too but hides certain options from the user.

Ubuntu users can choose either of these two desktop environments along with some more desktop environments as well.

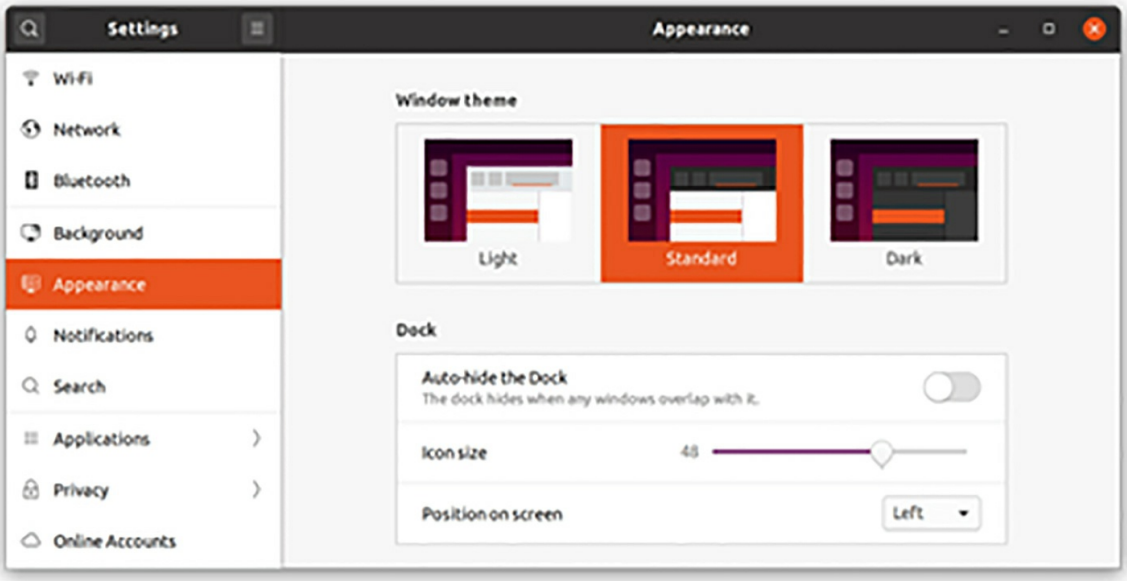
The default desktop environment in Ubuntu 20.04 is GNOME 3.36, but this

does not mean that you cannot install other desktop environments. Let us see what it has to offer to its users.

Theme

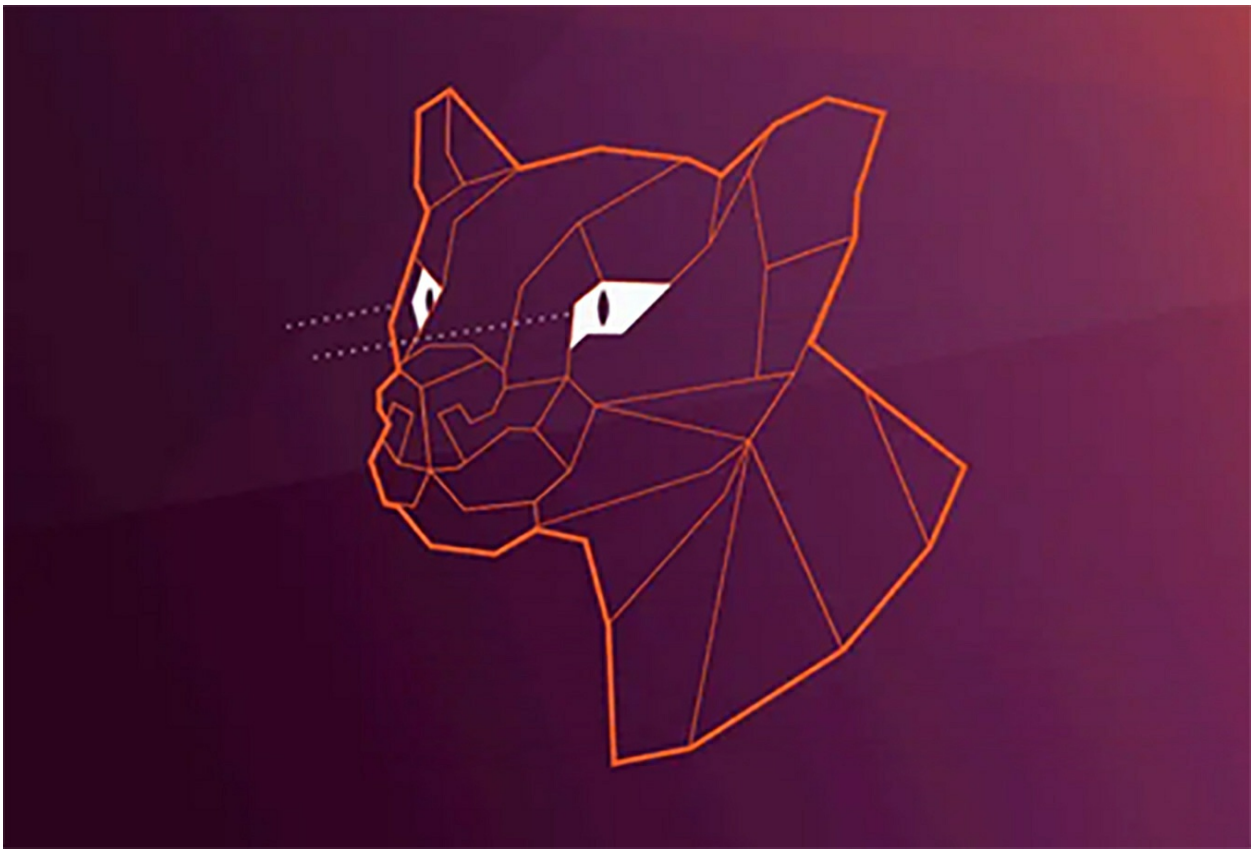
Ubuntu 20.04 LTS comes with a revamped Yaru theme, and you can see it right from boot up to the desktop. Developers of operating systems like to have a distinctive look for their operating system, which helps establish their brand's image. In January 2020, Canonical, the parent company of Ubuntu, had a design sprint in collaboration with the Yaru team and the Ubuntu design team. Yaru was introduced in Ubuntu 18.10.

Yaru has three types of variations, and it also has a sound theme. During the design sprint, the designers and developers also identified some UI improvements that could benefit the desktop. You can switch your desktop theme between Light, Standard, and Dark colors using the Appearance settings. You can also use the default sound alert that Yaru provides.



The revamped Yaru theme also lets you make changes to the boot splash screen and the installer windows. The Yaru team has now implemented the spinner that was available only on the desktop to appear even on the boot screen. If you have installed the system using the full-disk encryption option, then the boxes for the passphrase in the boot menu will match the desktop theme too.

Ubuntu has received a new animal mascot with each major release lately. This time also, a neatly designed creature has been integrated for the release. They have also given it a name. The Ubuntu 20.04 LTS mascot is called Felicity.



The default theme wallpaper features Felicity. The theme also has some more wallpaper in addition to the mascot wallpapers. All the images come from a copyright-free website, and every image is very focal.

GNOME 3.36

Ubuntu has defaulted to the GNOME desktop as the default desktop environment ever since the release of Ubuntu 17.10. The Ubuntu team has worked deeply with the GNOME developers and the Ubuntu community to create a solid GNOME experience for Ubuntu users. Ubuntu also has ties with the developers at Debian to keep updating GNOME to the latest packages.

GNOME 3.36 has changes that are visible to the user for a beautiful user

interface and hidden changes that improve the performance of the user interface and make the desktop experience very stable.

There is a new toggle for ‘do not disturb,’ which mutes all the notifications and helps users who want to stay focused on what they are doing. GNOME 3.36 also features simple and beautiful login and lock screens, resulting in a clean user interface. The login and lock screen blur the actual desktop background of the user.

The status menu features a quick suspend option. Also, users who like to be organized about the app grid can use a feature called ‘better app folder management’ with this release of GNOME.

Going by the tradition of every new GNOME release, desktop and system animations are now even smoother, and they do not put a lot of load on the CPU either.

Big Applications

Ubuntu now integrates easily with Microsoft Exchange and Google Gsuite. It ships with Firefox 75 as the default web browser that has continued features to protect a user's privacy.

The default email client on Ubuntu is Thunderbird that lets you access your emails from any provider quickly on your desktop. Irrespective of your email service provider, be it Gmail, Microsoft, other POP or IMAP provider, you can easily configure your email account on Thunderbird. Ubuntu 20.04 LTS comes with Thunderbird 68.7.0 preloaded with it.

You also get LibreOffice 6.4 for all your productivity needs. This is a free but powerful office suite that lets you perform any office related tasks with ease.

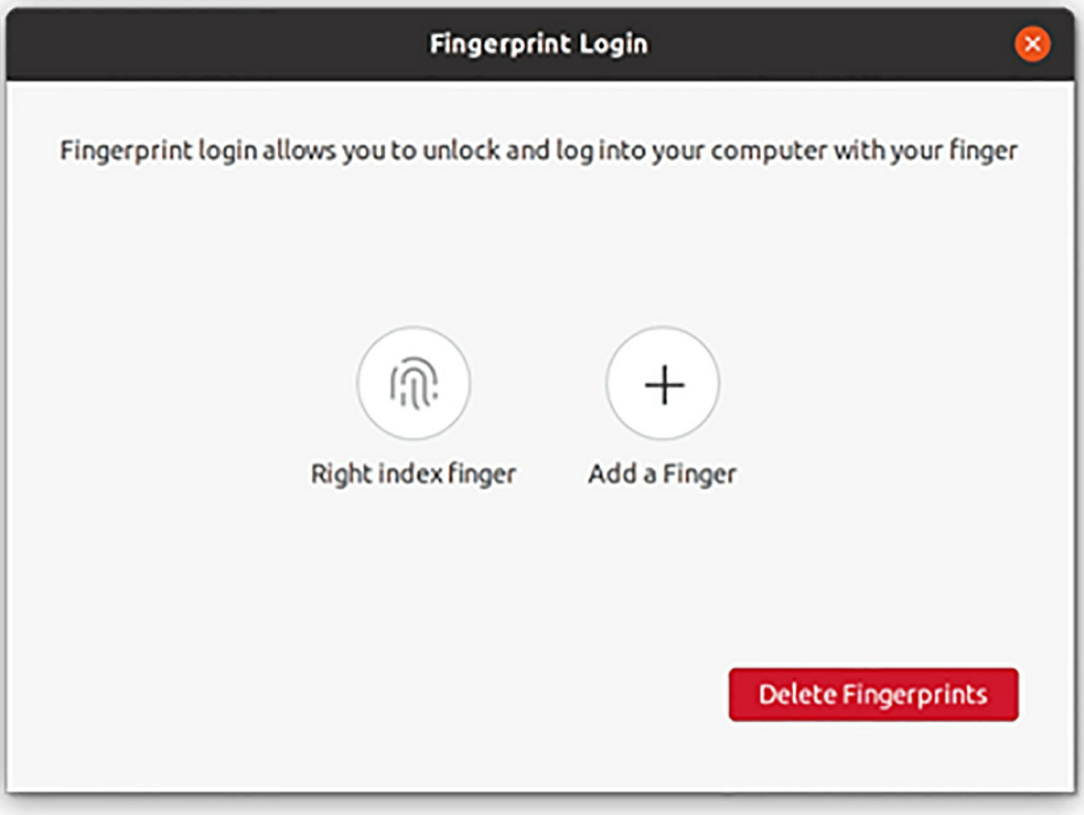
Tier 1 OEM Support

Ubuntu is a very popular operating system and is used on a large scale in schools, enterprises, government, and public sectors. The Ubuntu team works closely with computer manufacturers like HP, Dell, and Lenovo to meet pre-loaded hardware requirements from these areas. Let us go through some examples where Ubuntu works with OEM partners to add value to desktop users' Linux experience.

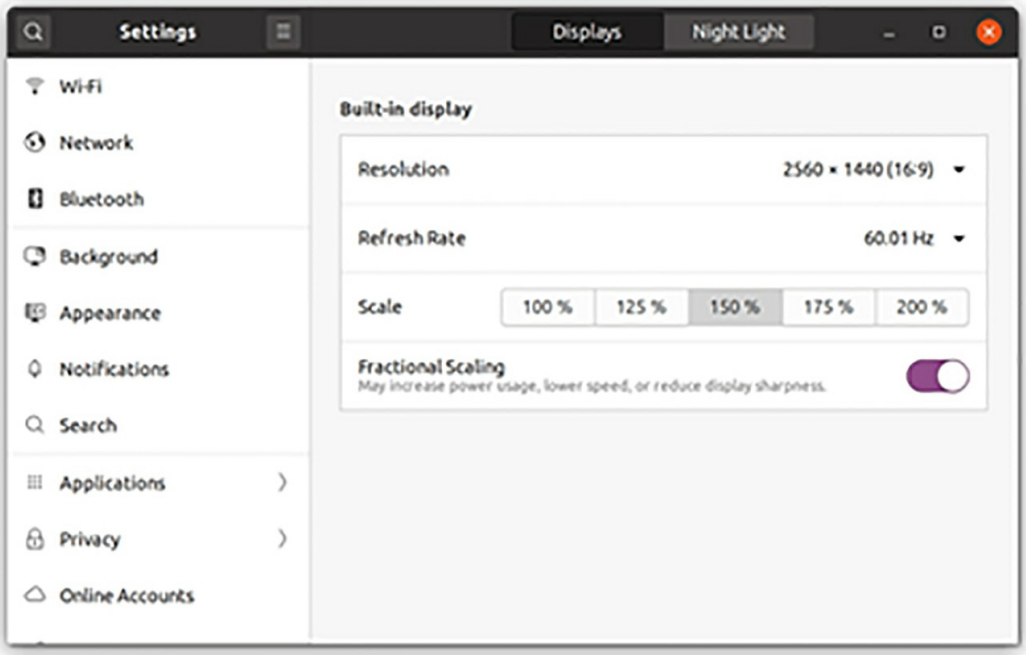
With the release of Ubuntu 20.04 LTS, you can get a certified device experience when you install the general release of Ubuntu. When you install Ubuntu on certified hardware from an OEM manufacturer, all the features of that specific hardware will be enabled automatically on the device, just as it would have been with factory images.

For example, if you have installed Ubuntu on a Dell device, the boot splash screen will automatically feature the Dell logo. Ubuntu also includes PulseAudio 14.0, BlueZ 5.53, and Sound Open Firmware to support SoundWire and SMIC, which is provided by OEM manufacturers on their hardware.

With security being critical, we can also see that many devices ship with a fingerprint reader and users prefer this way to unlock their systems to typing in a password. Ubuntu and the libfprint project are also making it possible to integrate this so that hardware vendors can be relieved for their devices that feature biometric authentication.



A feature called X11 fractional scaling has been a part of Ubuntu since Ubuntu 19.04 but was never exposed to users via the user interface. This has been exposed finally in Ubuntu 20.04 LTS so that Ubuntu users can enable fractional scaling easily through display settings. They can scale the display in increments of 25% from 100% to 200%.



Performance Computing and Gaming

Machine Learning and Artificial Intelligence are the latest in data engineering, and enterprises are adopting these technologies rapidly. Ubuntu has taken note of this and is already integrating support for Artificial Intelligence for enterprises, from developer workstations to server racks, to the Edge and the cloud.

Ubuntu empowers data science on desktops and servers by providing the latest applications, libraries, and drivers needed for it. Data scientists can use Kubeflow on powerful Ubuntu systems to create AI models before deploying them on servers or public clouds. Ubuntu has become a standard for machine learning, from Wall Street to Silicon Valley. It is the first choice for both startups and fortune 50 companies.

If you also have a GPU on your system that you use for entertainment and gaming, Ubuntu 20.04 LTS has new features that will make you very happy. Ubuntu 20.04 LTS has an updated Steam package that supports new controllers and virtual reality peripherals if you are a gamer. You will also have Feral Interactive's GameMode installed by default in Ubuntu. GameMode is a daemon that optimizes the operating system and the game process temporarily to have a rich gaming experience. At present,

GameMode optimizes the CPU governor, process niceness, Input-Output priority, kernel scheduler, GPU performance mode, screensaver inhibiting, GPU overclocking, and custom scripts.

If you have hybrid graphics for your system, you can now launch applications through the discrete GPU available in the GNOME shell. This can be done by using the ‘Launch on Discrete GPU’ item on the menu. This is supported for both NVIDIA and AMD GPUs.

Ubuntu also supports VA-API and nvenc through FFmpeg if you are a live streamer or a video content creator. These are features that are integrated into GPUs for encoding, decoding, filtering, etc. These features also offload intensives tasks from the CPU such that they are processed by the GPU instead. Applications such as Shotcut and OBG Studio use nvenc to benefit greatly from these hardware-encoding features.

ZFS and ZSYS

The ZFS file system was introduced as an experiment on Ubuntu 19.10. ZFS is shipped by default on Ubuntu 20.04 LTS that features encryption through hardware, pool trim, device removal, and improved performance. As mentioned earlier, ZFS is still in its experimental stages.

Zsys is an integration tool for Ubuntu and ZFS. Zsys takes an automatic snapshot of the system state whenever new software is installed, or the whole operating system is updated. This feature helps a user do a rollback on the system to a previous stable state if the new software or the new update causes issues in the system. You will be able to find these snapshots in the boot menu. This can also be used to create backups for the future.

Changes since Ubuntu 18.04 LTS

It is a known fact that Ubuntu users like to stick to Long Term Support releases. Users who are upgrading from Ubuntu 18.04 LTS to 20.04 LTS will see a huge number of changes as Ubuntu 20.04 gathers all the features that have been rolled through Ubuntu 18.10, 19.04, and 19.10. Let us highlight the most interesting changes.

GNOME Disks now support open-source disk encryption when you are formatting any disk. The performance for window previews and desktop

zoom has been improved. Crash reports are now sent automatically by the bug-reporting tool whenever an application crashes. This is helpful, as the user is not interrupted anymore while doing their work.

Support for microphones and speakers have been improved by tuning the sound panel in the GNOME Settings. The file search on the desktop has been improved by adding a tracker. Application switching has been made smoother by improving the Alt+Tab feature and the window preview feature. Unexpected errors on the system can be investigated with the introduction of the ‘Safe Graphics Mode.’

The CPU usage has been optimized, leading to better Desktop performance. The input and output latency has also been optimized for most graphics cards. This means that users will not enjoy faster and smoother frame rates. Ubuntu also allows DLNA sharing to share media to a smart TV or any other devices that support DLNA. Graphic drivers for NVIDIA GPUs are inbuilt into the Ubuntu install media now. This allows users to install drivers directly.

Ubuntu is a great operating system and what we have discussed in this chapter is just the cream of the case. We urge you to explore more features so that you customize the look and feel of Ubuntu as per your needs and enjoy the Ubuntu desktop to the fullest.

Chapter Five

Things to Do After Installing Ubuntu 20.04 LTS

The codename for Ubuntu 20.04 LTS is Focal Fossa. Ubuntu has always been proud about shipping the operating system with inbuilt applications that can help a user get started soon after installation. The default apps and settings are set as per what most people like. But we do not want to assume that you are most people. You may not want everything, or you may want more than the default offerings.

This chapter will help you tweak Ubuntu 20.04 LTS to improve your Ubuntu experience.

What's New?

Every version of Ubuntu differs from its previous release, and Ubuntu 20.04 LTS is no exception. The boot time on the Ubuntu 20.04 has been improved considerably, thanks to the new kernel compression algorithms. We have already discussed the Yaru theme and the OEM manufacturer's logo showing up on the boot splash as well.

The Dark Mode

The dark mode is the latest for any device today, be it a mobile or a computer. Everyone wants to have a dark mode on their devices, which is now possible with Ubuntu.

The dark mode does not default in Ubuntu, but you can easily switch to it. Follow the steps given below.

1. Open Settings > Appearance
2. Select 'Dark Windows' setting

That is it. You will see that the change is immediate, and everything is

darkened immediately, from backgrounds to toolbars. Even all the applications will not have a dark theme to them.

It is worth noting that the dark mode will not have any effect on the user interface of the GNOME shell in Ubuntu, such as calendar, notifications, and system menus.

If you get tired of the dark mode, you can always switch to the default theme again from the Appearance settings panel.

Tweaking GNOME

GNOME tweaks is an application through which you can customize your desktop even more.

You will get options to

- Move window buttons from the default right to left
- Change some themes
- Change the font and font size of your desktop
- Display weekdays on the clock
- Switch between workspaces
- Centre new windows automatically
- And more

Ubuntu tweak is an application through which you can fine-tune your desktop.

Launch Firefox on your Ubuntu system and go to the link apt://gnome-tweaks to install GNOME tweaks.

Get a Powerful Tool for File Previews

There is a tool known as GNOME Sushi that is a spacebar preview tool available for the GNOME shell desktop.

If you have been a Mac user, you will know what a spacebar preview means. Apple’s operating system was the first one to have this feature, and it became

trendy. You can select a file in the file manager and hit your spacebar to have a quick preview of that file.

Similarly, you can use Sushi to preview media, images, documents, get information about a file or a folder, and so on, without having to open a dedicated application to do it. Also, when you finally find the file that you want to open, Sushi gives you an option to do that.

Sushi is free and open-source, just like Ubuntu. You will be able to find it on the Ubuntu software app by searching for it by its name. Or you can simply launch Firefox again and type in the URL apt://gnome-sushi

Minimizing Applications

In Windows, when you click on any application on the taskbar, it gets minimized. This feature is not enabled by default on Ubuntu, but you can enable it.

This setting is not easily available via the user interface, but you can do it via the command line.

Simply open the Linux terminal and type the following command to enable minimize for Ubuntu.

> *gsettings set org.gnome.shell.extensions.dash-to-dock click-action 'minimize'*

This will be effective immediately. You can now click on any app on the dock, and you will see that it gets minimized.

Display Battery Percentage

This feature is useful if you have installed Ubuntu on a laptop and you want to see your battery percentage at a glance. It lets you know how much battery percentage is remaining without opening the status bar to do so.

You will need to use the GNOME tweaks app for this that we discussed above.

1. Launch GNOME Tweaks
2. Select Top Bar

3. Enable the Battery Percentage setting

You can also enable this via the command line by typing the following command on the terminal.

gsettings set org.gnome.desktop.interface show-battery-percentage true

Tweaking Touchpad Scroll

If you have installed Ubuntu on a laptop and are not comfortable with the default scroll of the touchpad known as 'Natural Scrolling,' you have the option to change it. You can change it such that the content moves in the direction of your scroll, that is, scroll down to move the page downwards.

1. Open Settings
2. Select Mouse & Touchpad
3. Toggle the Natural Scrolling option to On

That is it, and you should now be able to scroll comfortably as per your liking.

Setting up Livepatch

Livepatch is an option in Ubuntu that lets you have the Linux kernel updated automatically without the need to reboot your system.

This feature is aimed at servers, but it works just fine on desktops as well.

Launch the Livepatch application from the Application grid, and you're all set to install security updates to your Linux kernel without needing a reboot after installation.

Automatic Trash Deletion

There is an automatic trash deletion option hidden in the Privacy options in Settings. You can simply enable it so that trash is deleted at regular intervals. It is very useful to automatically free up space if you forget to do it manually from time to time.

Installing Amazing Software

You can find both free and paid software for Ubuntu. But which software would you need?

You can go through the Ubuntu software app and install software that you believe will suit your needs.

Don’t Do On Your Ubuntu System

There are a few things that are not advisable to do on your Ubuntu 20.04 LTS system. Let's go through them in brief.

Do not uninstall the default desktop

The default desktop environment in Ubuntu is the GNOME shell desktop. Not every user will like the desktop, but Ubuntu gives you the option to install other desktop environments alongside the default GNOME Shell, but uninstalling the default desktop environment after installing a new one is a poor decision. It can hamper system performance and result in irreversible damages if you uninstall your Ubuntu system's default desktop environment.

Do not run random commands

You will find millions of scripts for Linux all over the Internet claiming to do something amazing. While some of them are genuine, a majority of them can damage your system.

The thumb rule to follow is if you do not know the function of a particular Linux command, do not run it. This should be followed even strictly for commands that will perform multiple functions in one go. Always review the content of a script before you fire it on your terminal.

Chapter Six

The Linux Command Line

In this chapter, we will take you through the Linux command line. The Linux command line is similar to the command prompt in Windows and can be used to perform almost every task that you can do through a graphical interface by using the command line instead. By the end of this chapter, you would be proficient in managing your files and directories through the command line.

The Bash Shell

BASH, short for Bourne-Again Shell, is a Linux utility that helps you interact with the operating system through the command line. It is the first choice of shell on every Linux distribution, and naturally, you will find it on Ubuntu as well. The Bash shell can be accessed on Ubuntu through a utility called the terminal.

You can launch the terminal on Ubuntu by selecting the Application button on the lower-left corner of your desktop and then click on All at the bottom of the screen. Continue to select Utilities, and you will see all the system tools under Utilities. Click on Terminal to launch it.

Upon launching the bash shell, you will see a string that indicates that input is expected from the user. This string is called shell prompt. If you are the root user or the superuser for your Ubuntu system, the prompt will end with a # sign. If you are a non-root user, the prompt will end with a $ sign.

[root@desktop ~]#

[student@desktop ~]$

As mentioned, the Linux Bash shell is just like the command prompt available in Windows, but both use a different scripting language, and the scripting language in the Bash shell is much superior as compared to the one available in the Windows PowerShell utility. You can automate numerous

tasks in the Linux operating system using the Bash shell. You can also use the Bash shell to perform tasks that would be rather complicated, even on the graphical interface.

The Bash shell, as mentioned earlier, can be accessed through the terminal in Ubuntu. The keyboard works as the input device for the terminal, and the monitor works as the output device to show you the result of your commands. There are virtual consoles available on Linux to access the Bash shell as well. This helps you to have multiple virtual consoles on a single physical machine, and multiple users can log in via the multiple virtual consoles.

Shell Basics

There are three parts to the command that you enter on the shell prompt in Linux.

1. *Command that you want to run*
2. *Options that will define the behavior of the command*
3. *Arguments that are the command's targets*

The command defines the program that you want to execute. The command can be followed by options or no options at all. The options will define how a command will behave and how it will function. You can use a dash or two dashes to specify an option. The dashes are appended to options so that options can be distinguished from arguments.

Example -a or --all

Arguments also follow the command, and you can have a single argument or multiple arguments. Arguments are the targets on which the command gets executed upon.

A basic command looks like this.

usermod -L John

The command in this example is *usermod*

The option is *-L*

The argument is *John*

This command is used to lock the password of the user John on the Linux system.

It is useful to know which options go with which commands to be efficient on the command line. The *--help* option with any command will give you a list of options that can be used with that particular command. It is not important to know every option by heart. The list also gives you a description of what the option does.

Let us see an example of this with the **grep** command. The grep command is used to search through a string or a file. If the file has a string that matches the one specified by you in your grep command, the output shows every line of the file that contains the specified string.

[student@desktop ~]$ grep --help

Usage: grep [OPTION]... PATTERN [FILE]...

Search for PATTERN in each FILE or standard input.

PATTERN is, by default, a basic regular expression (BRE).

Example: grep -i 'hello world' menu.h main.c

Regexp Selection and Interpretation:

-E, --extended-regexp PATTERN is an extended regular expression (ERE)

-F, --fixed-strings PATTERN is a set of newline-separated fixed strings

-G, --basic-regexp PATTERN is a basic regular expression (BRE)

-P, --perl-regexp PATTERN is a Perl regular expression

-e, --regexp=PATTERN use PATTERN for matching

-f, --file=FILE obtain PATTERN from FILE

-i, --ignore-case ignore case distinctions

-w, --word-regexp force PATTERN to match only whole words

-x, --line-regexp force PATTERN to match only whole lines

-z, --null-data a data line ends in 0 byte, not newline

Miscellaneous:

-s, --no-messages suppress error messages

-v, --invert-match select non-matching lines

-V, --version display version information and exit

--help display this help text and exit

Output Control:

-m, --max-count=NUM stop after NUM matches

-b, --byte-offset print the byte offset with output lines

-n, --line-number print line number with output lines

--line-buffered flush output on every line

-H, --with-filename print the file name for each match

-h, --no-filename suppress the file name prefix on output

--label=LABEL use LABEL as the standard input file name prefix

-o, --only-matching show only the part of a line matching PATTERN

-q, --quiet, --silent suppress all normal output

--binary-files=TYPE assume that binary files are TYPE;

TYPE is 'binary', 'text', or 'without-match'

-a, --text equivalent to --binary-files=text

-I equivalent to --binary-files=without-match

-d, --directories=ACTION how to handle directories;

ACTION is 'read', 'recurse', or 'skip'

-D, --devices=ACTION how to handle devices, FIFOs, and sockets;

ACTION is 'read' or 'skip'

-r, --recursive like --directories=recurse

-R, --dereference-recursive

likewise, but follow all symlinks

--include=FILE_PATTERN

search only files that match FILE_PATTERN

--exclude=FILE_PATTERN

skip files and directories matching FILE_PATTERN

--exclude-from=FILE skip files matching any file pattern from FILE

--exclude-dir=PATTERN directories that match PATTERN will be skipped.

-L, --files-without-match print only names of FILEs containing no match

-l, --files-with-matches print only names of FILEs containing matches

-c, --count print only a count of matching lines per FILE

-T, --initial-tab make tabs line up (if needed)

-Z, --null print 0 byte after FILE name

Context Control:

-B, --before-context=NUM print NUM lines of leading context

-A, --after-context=NUM print NUM lines of trailing context

-C, --context=NUM print NUM lines of output context

-NUM same as --context=NUM

--group-separator=SEP use SEP as a group separator

--no-group-separator use empty string as a group separator

--color[=WHEN],

--colour[=WHEN] use markers to highlight the matching strings;

WHEN is 'always', 'never', or 'auto'

-U, --binary do not strip CR characters at EOL (MSDOS/Windows)

-u, --unix-byte-offsets report offsets as if CRs were not there

(MSDOS/Windows)

'egrep' means 'grep -E'. 'fgrep' means 'grep -F'.

Direct invocation as either 'egrep' or 'fgrep' is deprecated.

When FILE is -, read standard input. With no FILE, read . if a command-line

-r is given, - otherwise. If fewer than two FILEs are given, assume -h.

Exit status is 0 if any line is selected, 1 otherwise;

if any error occurs and -q is not given, the exit status is 2.

Report Bugs to: bug-grep@gnu.org

GNU Grep home page: <http://www.gnu.org/software/grep/>

General help using GNU software: <http://www.gnu.org/gethelp/>

You may find the command line a bit difficult to understand initially, but you will enjoy it more than the graphical interface once you get used to the command line.

Let us try to understand the essential syntax of the command line.

- Options are wrapped using square brackets []
- If a command is followed by …, it indicates the list of items belonging to the same type
- If you have specified multiple items and separated them using a pipe | it indicates that only of them should be used
- Variables are specified using angle brackets <>. If you see <filename> in the syntax, you know you have to replace it with the actual filename

Let us go through an example.

```
[student@desktop ~]$ date --help
date [OPTION]... [+FORMAT]
```

This indicates that the command is **date,** and it takes the options represented by **[OPTION]**. It also takes another option **[FORMAT]** prefixed with a + sign.

Executing Commands

The bash shell's primary function is to interpret the commands input by a user and convert them into system-specific instructions for the Linux operating system. We already know that a string you end on the shell prompt is made up of three parts, command, options, and arguments. Every word that is entered on the command prompt needs to be separated by using a blank space. The Linux system already has a script file where the function of the command you type is defined. You can manipulate this script by passing options and arguments to your command.

Let us go through the most common commands.

The **date** command will display the current date and time that has been set on your Linux system. You can also use the same command to set a new date and time if required. The command can be followed up with a + sign as an argument if you want to display the date and time in a specific format.

[student@desktop ~]$ date

Sat Aug 5 08:15:30 GMT 2019

[student@desktop ~]$ date +%R

08:15

[student@desktop ~]$ date +%x

08/05/2019

The **passwd** command is used to change the password for a user in the Linux system. The command will request you to enter the user's existing password before it allows you to set a new password. The Linux system expects a strong password comprising letters in upper and lower case, numbers, and symbols. Moreover, the password cannot be a dictionary word. A non-root user can only change their password through the command like. A root user has the right to change any user's password through the command line.

[student@desktop ~]$ passwd

Changing password for user student.

Changing password for student.

(current) UNIX password: type old password here

New password: Specify a new password here

Retype new password: Type new password again

passwd: all authentication tokens updated successfully.

As opposed to file formats in Windows, files in the Linux file system are not specified using any extensions. You can still use the **file** command if you

wish to know the format of any file. The file needs to be passed along with the command as an argument.

[student@desktop ~]$ file /etc/passwd

/etc/passwd: ASCII text

If you pass a directory instead of a file to this command, the output will tell you that it is a directory.

[student@desktop ~]$ file /home

/home: directory

You can use the **head** and **tail** command to print the first 10 lines or the last 10 lines of a file, respectively. These commands also take the **-n** option that can be used to specify a custom number of lines to be output.

student@desktop ~]$ head /etc/passwd

This command will print the first 10 lines of the passwd file.

[student@desktop ~]$ tail - n4 /etc/passwd

This command will print the last four lines of the passwd file.

You can pass a file as an argument with the **wc** command to get the number of lines, words, and characters in a file. The command supports the **-l, -w, -c** options that represent lines, words, and characters, respectively.

[student@desktop ~]$ wc /etc/passwd

40 80 2000 /etc/passwd

This shows that the passwd file has 40 lines, 80 words, and 2000 characters.

As you can see, if you do not specify any option with the wc command, the output shows everything. If you pass a particular option with the command, it will only show the output for only that particular parameter.

The **history** command will show you all the commands that you have typed in the past. These commands will also have a number next to them. You can then type ! with the command number to know the complete command that

was typed by you.

[student@desktop ~]$ history

1 clear

2 who

3 pwd

[student@desktop ~]$!3

/home/student

You can see that !3 expanded the entire command for pwd and even showed the output for the user's present working directory.

You can use the keyboard arrow keys to navigate through the output of the history command. The up arrow key takes you to the commands on top, while the down arrow key takes you to the commands below. The left and right arrow keys will help you edit the command that you are on at present.

File Management Through the Command Line

This section will go through some important commands that will help you manage files and directories on Linux. You will learn commands that will let you create, delete, copy, move, and organize files and directories.

Linux File System Hierarchy

To understand the Linux file system properly, you need to understand the Linux file system hierarchy first. The Linux file system hierarchy visually looks like an inverted tree with the root at the top and then branching downwards to form other parts.

The root directory is represented as **/** and is located at the top of the file system hierarchy. The / character is also used to represent file paths in Linux. For instance, **var** is a subdirectory under the root directory and is represented as /var. Similarly, there is a subdirectory called **log** under the var directory, and its path is represented as /var/log.

The root directory has a particular set of subdirectories under it that store

specific files. For instance, the**/boot** subdirectory will have files that are responsible for the boot process of the Linux system.

Let us go through the main subdirectories under the root directory.

/usr

This subdirectory has files of software that is common to all users. It is further subdivided as follows.

/usr/bin: User command files

/usr/sbin: Commands used in system administration

/usr/local: Files of software that has been customized locally

/etc

Files related to system configuration are stored in this subdirectory.

/var

This subdirectory stores files that change dynamically, such as databases, logs, etc.

/run

Certain files that get created during runtime are stored in this subdirectory. These files will get recreated again on the subsequent boot.

/home

This subdirectory has all the different users of the Linux system listed under it. Every user has their particular home directory as well. A user can store all their data under their home directory. A user cannot access the home directory of another user.

/root

This is the home directory for the root user, and no one other than the root user can access it.

/tmp

As the name suggests, this subdirectory stores all temporary files. If files in this directory are older than ten days and haven’t been accessed, the system deletes them automatically. A similar directory exists at /var/tmp that has

temporary files too. The files here, if not accessed in the last 30 days, get deleted automatically.

/dev

Hardware devices on the Linux system are stored as files under this subdirectory.

File and Directory Management

Managing a file or a directory on the Linux operating system refers to creating, modifying, and deleting files or directories. Let us go through the common commands that are used to manage files and directories in Linux.

Activity	**Single Source**	**Multiple Source**
Copy file	cp file1 file2	cp file1 file2 file3 dir
Move file	mv file1 file2	mv file1 file2 file3 dir
Delete file	rm file1	rm -f file1 file2 file3
Create directory	mkdir dir	mkdir -p par1/par2/dir
Copy directory	cp -r dir1 dir2	cp -r dir1 dir2 dir3 dir4
Move directory	mv dir1 dir2	mv dir1 dir2 dir3 dir4
Delete directory	rm -r dir1	rm -rf dir1 dir2 dir3

mv file1 file2

This command renames file1 to file2

cp -r dir1 dir2

This command will copy the contents of dir1 to dir2

rm -r dir1

This command will delete the contents of dir1

-r is used to process the source directory recursively

mv dir1 dir2

This command will copy the contents of dir1 to dir2 if dir2 exists. If dir2 does not exist, dir1 will be renamed to dir2

cp file1 file2 file3 dir

mv file1 file2 file3 dir

cp -r dir1 dir2 dir3 dir4

mv dir1 dir2 dir3 dir4

Files or directory contents are copied or moved to the directory specified at the end.

rm -f file1 file2 file3

rm -rf dir1 dir2 dir3

This command deleted the files or directories. Kindly use this carefully as the -f uses a force option will delete everything without any confirmation prompt

mkdir -p par1/par2/dir

The command will create new directories. Kindly use this carefully as using -p will keep creating directories starting from the parent and irrespective of typing errors

Let us see through examples how these commands work.

Directory Creation

You can use the **mkdir** command to create directories and subdirectories. The directory should be created without issue as long as the specified name does not exist or the parent directory is specified properly. If not, then you will get an error. If you use the **-p** option while creating directories, the parent directory will get created if it does not exist. You need to be careful with this

option since it does not check for any spelling errors.

[student@desktop ~]$ mkdir Drawer

[student@desktop ~]$ ls

Drawer

You can see that the command creates a new directory called Drawer in the student user's home directory.

[student@desktop ~]$ mkdir -p Thesis/Chapter1

[student@desktop ~]$ ls -R

Thesis thesis_chapter1

When we use the option -p with the mkdir command, the parent directory Thesis, and the subdirectory Chapter1 are created simultaneously.

Copying Files

Files can be copied on the Linux system by using the **cp** command on the command line. The command can be used to copy one or more files, and you can copy files within the same directory or from one directory to another.

You will need to specify a unique destination file. If the file already exists, the copy action will overwrite the existing file with the source file's contents.

[student@desktop ~]$ cd Documents

[student@desktop Documents]$ cp one.txt two.txt

In the above example, contents of the file one.txt are copied to the file two.txt

[student@desktop ~]$ cp one.txt Documents/two.txt

In the above example, contents of the file one.txt are copied to the file two.txt that is under the Documents directory.

Moving Files

You can use the **mv** command to move files from one place to another. The

mv command has an alternate function as well. If you are using the mv command within the same directory, it will merely perform a rename operation on the source file. If you are specifying the destination as a different directory, the source file will be moved under the destination directory.

The move operation from one directory to another may take more time if the file size is huge.

[student@desktop ~]$ ls

Hello.txt

[student@desktop ~]$ mv Hello.txt Bye.txt

[student@desktop ~]$ ls

Bye.txt

The mv operation is taking place within the same directory. The file is just renamed from Hello.txt to Bye.txt

[student@desktop ~]$ ls

Hello.txt

[student@desktop ~]$ mv Hello.txt Documents

[student@desktop ~]$ ls Documents

Hello.txt

The mv operation is not taking place within the same directory. The file is moved from Hello.txt to under the Documents directory.

Deleting Files and Directories

The **rm** command is used to delete files and directories in Linux. Combine is with **-r,** and it will delete everything recursively in the specified path.

Note that Linux does not have a concept like Recycle Bin in Windows. So when you delete something in Linux, it is deleted permanently. There is no way of restoring it again.

[student@desktop ~]$ ls

File1.txt Directory1

[student@desktop ~]$ rm file1

[student@desktop ~]$ ls

Directory1

[student@desktop ~]$rm -r Directory1

[student@desktop ~]$ ls

[student@desktop ~]$

You can see how the rm and rm -r commands can be used to delete files and directories in the above example.

Also, note that if a directory does not contain any files, you can delete it using the **rmdir** command as well.

Chapter Seven

Editing Text Files in Linux Using Vim

In this chapter, we will learn how you can edit text files in Linux. There are applications available on the graphical interface of any Linux distribution that you can use to edit text files, but the essence of editing a text file in Linux is via the command line. The habit of editing text files in Linux through the command line will take you a long way, and we personally advise that this is the best way to edit files in Linux.

There are many text editors available in Linux, such as nano, vi, and vim. We will be using the Vim text editor for our book, as it is the most popular one. If you have installed Ubuntu on your computer as per the installation chapter, vim may or may not be available by default on it. But there's nothing to worry about. Installing vim on Ubuntu is very easy.

Launch the terminal on your Ubuntu system and type the following command.

apt install vim

Remember that you need to be logged in as the root user to run this command.

Hit enter, and the system will install vim on your Ubuntu system.

Introduction to Vim

Editing files is an important skill in Linux as a beginner and as you progress with your experience with Linux systems. If you start loving Linux as an operating system and want to make a career in it, you may want to look up a Linux System Administrator profile. Linux system admins use text editors like vim daily to edit configuration files in Linux and write their own shell scripts to automate tasks.

Vim is an extensive tool. It supports text completion modes scripts in multiple programming languages, file type plugins, and many other features. Vim is so popular that third-party plugins are available for it on the internet that can help you achieve many things, such as editing a simple file, auto completion for various programming languages, and very simple tasks such as to-do lists.

Vim Versions

There are three versions of vim, and every version has its use case, but it is possible to have all three versions side by side and use all of them. The following are the versions of vim.

Vim-Minimal

This package contains an older version of vim known as just vi. The commands that can be used with this version of vim are also those that could only be used with vi.

Vim-Enhanced

This package contains vim and includes syntax highlighting, spell check, and file extension plugins.

vim-X11

This package of vim includes gvim that has graphical interface support for editing files too. The most popular feature of gvim is its menu bar. It is popular with new learners who struggle initially to remember vim commands on the terminal. It also has support for the mouse to be used inside a vim session on the terminal too.

Vim Modes

Vim is not known to be a very easy editor because it has multiple modes. This means that the keyboard keys you use have different functions based on the mode that you're in. You do not have to worry as this chapter will guide you step by step with vim, and you will fall in love with it.

The following three modes are present in vim.

Function	Mode
Command mode	You can do file navigation, cut, copy and paste functions, and other simple commands in this mode. Commands like undo and redo can also be performed in this mode
Insert mode	Normal text editing can be performed in this mode. There is a variation of insert mode known as replace mode, in which you can replace text instead of inserting it
Ex mode	Functions such as opening a file, saving it, or quitting vim can be performed in this mode. It also supports a few other complex functions. The output of programs can be inserted into the current file using this mode. Vim configurations can be done from this mode as well. Everything that one can perform in ex can be done in this mode of vim

Vim Workflow

This section will teach you how to open text files using vim, insert and replace text, move the cursor within vim, save files, and look at help options.

Vim Editor Basics

No matter the text editor you choose to use in the long run, it should be able to help you perform these three primary tasks.

1. Create a new file or open an existing file
2. Modify the contents of the file
3. Save the changes and exit

Opening a File in Vim

You can easily open files in vim on the terminal by specifying the file name as an argument to the **vim** command. For example, if you wish to open the file, you can just type the command below.

[student@desktop ~]$ vim filename

Note that if the file specified by you in the vim command does not exist, vim will end up creating a new file by that name instead and take you right to the editor.

By default, vim launches in the command mode. You will see information such as the file name, number of characters, number of lines, etc., in the bottom left corner of your vim editor. You will also see information about what part of the text file is currently visible on the screen, such as All for all, Top of the line at the top, Bot for the last lines in the file, or you may see a percentage indicating the part of the file you're currently at. In vim terminology, the last line is known as the ruler.

Editing Text in Vim

When you are in the command mode of vim, your keyboard keys will not exactly do what you intend to. This is because inserting text is not supported in command mode, and it is to be used to specify some functions to vim, such as cursor movement or copy and paste.

You can switch to insert mode from the command mode in vim by using commands in the command mode that correspond to various functions. Let's have a look at the commands.

Key	**Result**
i	This will switch you to the insert mode, and you can start typing before the current position of the cursor
a	This will switch you to insert mode as well but will start typing after the current position of the cursor
I	Will move the cursor to the beginning of the current line and switch you to insert mode
A	Will move the cursor to the end of the current line and switch you to

	insert mode
R	Will switch you replace mode, starting at the character where the cursor currently is. You cannot insert additional characters in the replace mode, but every character you type replaces the given document's current character.
o	A new line is opened below the current line, and you then switch to insert mode
O	A new line is opened above the current line, and you then switch to insert mode

You will know if you are in the insert or replace mode of Vim from the indicator at the bottom that shows **--INSERT--** or **--REPLACE--**

To return to the command mode, simply press the Escape key on your keyboard.

You can use keyboard keys in the command mode to navigate around your file. The following table will show the most common keys.

Key	**Result**
h	The cursor is moved to the left by one position
j	The cursor is moved down by one line
k	The cursor is moved up by one line
l	The cursor is moved to the right by one position
^	The cursor is moved to the start of the current line

$	The cursor is moved to the end of the current line
gg	The cursor is moved to the first line of the document
G	The cursor is moved to the last line of the document

Note that you can press the escape key on the keyboard to cancel the current command on return to vim's command mode. It is a good practice to press the escape key twice to ensure that you have returned to the command mode.

Saving Files in Vim

You can save files in vim from the ex mode. You can enter the ex mode by pressing the colon (:) key while you are in the vim command mode. So if you are in the insert mode of vim, enter escape to first come to the command mode and then press the colon key to go to the ex mode. You will see a colon on vim's ruler when you are in the ex mode that indicates you being in the ex mode. You can type a command in the ex mode and then press enter to complete the command.

The following table gives you a list of commands that can be used in vim's ex mode.

Command	**Result**
:wq	Save the current file and quit vim
:x	Save the current file if there are no unsaved changes, and then quit vim
:w	Save the current file and stay in the editor. Does not quit vim
:w <filename>	Use a different filename to save the current file

:q	Quit the current file ONLY if there are no unsaved changes
:q!	Ignore any unsaved changes and quit the current file

w stands for write in the above table, which means write changes to the file and save it. q stands for quit, and the exclamation mark is to force a command.

Getting Help in Vim

Vim has extensive online help that can be accessed through vim itself. You can type :help from the command mode in vim, and the help screen will pop up with a lot of help documentation. You can type **:help subject** if you want to find help for a particular topic.

The help screen will launch in a new split window. You can also type the command **vimtutor** while you are in the command mode of vim. This will launch a guided tour of vim, which is helpful for a new user.

Editing in Vim

In this section, we will go through vim shortcuts that are very helpful while editing. They are also helpful while performing tasks such as copy and paste, replacing text, etc. We will also learn about the undo and redo options in vim.

Movement in Vim

Most Linux editors have restrictions where you can move your cursor only through a single character or a single line. These options are also available through the command mode in vim, but vim provides additional options to navigate through the entire document with ease. These vim options will let you move through a text file in vim per word, sentence, and paragraphs. Note that these options will only work in the command mode of vim and will not work in the insert mode whatsoever.

Key	**Function**
w	The cursor is moved to the start of the next word

b	The cursor is moved to the start of the previous word
(	The cursor is moved to the start of the current sentence or the previous sentence
)	The cursor is moved to the start of the next sentence
{	The cursor is moved to the start of the current paragraph or the precious paragraph
}	The cursor is moved to the start of the next paragraph

The commands mentioned above also take numbers as a prefix. For example, if you type 5w, it will move the cursor after the next five words. Similarly, if you type 13, the cursor will move to the start of a line after 13 lines. In vim terminology, this is known as count.

Text Replacement

You can use the **change** command in vim to replace one or more words in the text document. You can execute the change command on vim by pressing the **c** key on your keyboard and then following it up with the required cursor movement. For instance, if you enter **cw** on the keyboard, the cursor will move from its current position to the end of the word that you were at. Vim will switch to the insert mode, and the text you want to delete is replaced.

You can also use a few shortcuts to perform editing with more ease.

- If you enter **cc**, that is, the c key twice, the replacement function is applied to an entire line in your text document. So you can replace an entire line or multiple lines if you prefix this command with a number. This applies to other functions as well, such as deleting a line.
- The movement commands can be prefixed with **i** or **a** to manipulate your movement. For instance, if you use the command **ciw**, you can replace the entire word but not just from

the current position you're at. You can replace that work from the entire document. Similarly, you can use **c$** to replace everything till the end of a line. This method also comes in handy in case you want to delete something.

- If you want to select and replace a single character under your cursor, you can use **r**.
- You can use **~** to change the case of the character under your cursor.

Text Deletion

Text deletion in vim world is similar to text replacement in vim. You can use the **d** command to delete text in vim, and the movement commands that apply to text replacement also work for text deletion. You can also use the command **D** to delete an entire line of text from the point where the cursor is. You can use the **x** command to delete a single character under the cursor.

Copy and Paste

The copy and paste terminologies in vim differ from the usual terms used around on most text editors. The copy operation is known as yank, and the paste operation is known as put. The keyboard keys assigned for these operations reflect this as the command for yank is **y,** and the command for put is **p** or **P**.

You can use the methods used with replace and delete for yank as well. You can also prefix a number to the yank operation. For instance, **5aw** will copy the current word and the next five words. You can copy an entire line by using the **yy** command.

You can use the **p** or **P** command to paste something. When you use the lower case for p, the content is pasted after the cursor or under the current line. When you use an upper-case P, the content gets pasted before the cursor or above the current line. As with all commands, you can also add a numeric prefix to the put commands to perform puts of multiple lines.

Multiple Registers in Vim

A register refers to a clipboard that stores copied content. Most text editors have only one clipboard, but vim has 26 clipboards or registers. There are a

few special registers in vim too. The presence of multiple registers makes copy-paste operations very easy for the user as they never have to worry about losing their data or moving the cursor too much. The user can specify the register they want to use. If a register is not specified, vim will use an **unnamed** register by default. The normal registers are named between **a** to **z** and can be used by using the "**registername** syntax between the count for a command and the actual command that you wish to use. For instance, if you want to copy the current line and the next four lines into the register at p, you can use the command **4"pyy**.

If you wish to paste the content stored on a particular register to the document, you can use the registername before the p command. For instance, if you want to paste the content from the register b, the command would be **bp**.

When you store something to a named register, it automatically gets stored to the unnamed register too.

It is possible to use the vim registers for the delete and replace operations too. Again, if you do not specify any named register, vim will use the default unnamed register. Using the uppercase version of a register results in the text getting appended rather than being replaced.

Special Registers

Vim has a set of 10 special registers numbered from 0 to 9. The most recently copied text is stored in 0, and the most recently deleted text is stored in 1.

The content stored in named registers is retained across all sessions, but content stored in special registers gets erased once you leave the session.

Visual Mode in vim

There is also support for a visual mode in vim, which saves you from the constant worry of counting the number of characters, words, lines, or sentences. When you are in the visual mode, it will be indicated as **--VISUAL--** on the vim ruler. The visual mode lets you select text by moving the cursor around. While you are in the visual mode of vim, cursor movement is not needed to use yank, replace, or delete commands. It gets applied automatically to the text that is selected by the cursor.

The visual mode of vim has three flavors.

1. Character-based visual mode, which starts with **v**
2. Line-based visual mode, which starts with **V**
3. Block-based visual mode, which starts with **Ctrl+V**

You can also use the mouse to select text while you are in the visual mode of vim.

Searching in Vim

You can search for something in a document using two ways in vim.

1. If you press the / key, the search will search after the cursor position in the current document
2. If you press the ? key, the search will scan through the document behind the current position of the cursor

While in the search mode, you can use a regular expression to search something in the document to find a match. You can use the **n** and **N** keys on the keyboard to navigate the previous match and the next match.

There is a shortcut that will help you search for a word that is currently under your cursor. You will simply need to use the * key to find the next occurrence of that same word in the document.

Search and Replace in Vim

You can implement search and replace in vim through its ex mode. You can implement a search and replace for regular expressions such as patterns, ranges, flags, strings, etc.

The range can be a line number or a range of line numbers. It can also be a simple search term that will match the current document's lines. The search and replace functions operate only on the current line or the highlighted line in the vim's visual mode.

Example:

Consider a document that has the word **cat** at multiple points throughout the

document. You want to replace this word with the word **dog** instead. You are not bothered about the case and want it to be applied only where cat is an independent word. You can use the following command in this case.

:%s/\<cat\>/dog/gi

Undo and Redo

Vim, just like any text editor, supports undo and redo operations. You can undo an action by pressing the **u** key in the command mode, and you can redo an action by pressing the **Ctrl+r** command in the command mode.

Chapter Eight

User and Groups in Linux

This chapter will introduce you to users and groups in Linux and the password policies for them. By the end of this chapter, you will know everything about user and group roles in a Linux system and how the operating system uses them.

Users and Groups

We will learn about users and groups in this section and try to understand how they are associated with the Linux operating system.

Who is a User?

Every program or process that is running on the Linux operating system runs as a user. A user is the owner of a file or a directory in the operating system. Access and restrictions can be created on a file or a directory based on the user. This being said, when a process is running as a user, the files and directories that a particular process has access to will depend upon the user of that process.

You can type the **id** command on the Linux terminal to know the user's details that are currently logged into the operating system. You can pass another username as an argument to the id command if you wish to get another user's basic details.

You can use the **ls -l** command to know the user associated with a file or a directory. This user will be the owner of that file or directory.

You can use the **ps** command to know the ongoing processes in the Linux operating system. The default output will give you a list of all the ongoing processes of the current shell only, but you can list the ongoing processes across every shell by using the **ps a** command. You can also pass the **u** option with the ps command if you want to know the user that is associated with a particular process. The first column of the output will show you the user that

owns the process.

The outputs that you just learned about give you the user details by their usernames, but the Linux system tracks all its users using a user ID known as UID. The Linux system maintains a database to map the username to the numeric IDs. The information for all users is stored in a file at /etc/passwd. This file has seven fields as follows.

username: password: UID: GID: GECOS: /home/dir: shell

- The username field will have the username for a user in the Linux system.
- The password is the password for that user, but it is not stored here anymore. It is now stored in a file at /etc/shadow in a hashed format.
- The numeric ID assigned to a user in the Linux system is mentioned under UID.
- The group ID for a user is mentioned under GID. We will discuss groups later in this chapter.
- The GECOS field uses arbitrary text and is mostly the complete name for the user.
- The user's home directory in the Linux system is mentioned under /home/dir and is the directory where all the files for a user are stored.
- The shell field mentions the program that runs when the user logs in. For a regular user, it will be the command prompt.

What is a Group?

The Linux system maintains groups just like users. Groups are maintained using numbers, too, known as Group IDs or GIDs in short. The information for groups is stored in the file at /etc/group.

There are two types of groups, primary and supplementary.

Primary Group

- Every user is associated with exactly one primary group
- The primary group for a user is defined in the 4th field of the file /etc/passwd as listed by the GID field
- The primary group is the group owner of any new file that is created by a user
- The primary group number by default for a user is the same number as the user. The primary group is also known as the User Private Group (UPG) as it has only the user as a member of it

Supplementary Group

- A user can be a part of zero or more supplementary groups
- Supplementary groups exist so that a user is not restricted to one group, and resources can be allocated to users based on their groups

The Superuser

This section is about the superuser of a Linux system commonly known as the root user.

The Root User

Every operating system has one user that is known as the superuser with all access and rights to the operating system. If you have used a Windows operating system before, you may have heard about the superuser known as the administrator. Similarly, in Linux operating systems, the superuser is known as the **root** user. The root user has complete privilege to do anything in the Linux system and is usually used to manage the system. The root user can only perform tasks that modify the operating system for everyone such as installing or uninstalling software, managing files and directories, etc.

A root user can only manage additional devices attached to an operating system. There are certain exceptions, such as USB drives that can be managed by a regular user as well. A regular user can add or remove files

from a USB drive, but if you want to do the same for a fixed hard drive, you'd need to be logged in as a root user.

You may have heard the saying, "with great power comes great responsibility." This applies to the root user as well. The root user has so much privilege and access that it can damage the entire system if misused. The root user can add or remove files, make changes to user accounts, create loopholes in the system, etc. If the root user's password is compromised, some else can take over the system. The thumb rule in Linux says that you should always log in as a ruler user and escalate tasks to the root user when required.

The su command

The **su** command in Linux is used to switch to another user account. You need to pass the username as an argument to the su command to let the system know which account you wish to switch to. If you do not pass any argument, the system will switch you to the root account. If you are a regular user trying to switch to another user account, you will be asked to enter the password for that user; but if you are logged in as the root user and try to switch to another user account, you will not be asked for any password.

su - <username>

[student@desktop ~]$ su -

Password: rootpassword

[root@desktop ~]#

You will be logged into a nologin shell session if you type the command **su username**. If you need a login shell for the new user you are switching to, you will need to use the hyphen - sign along with your command, as shown in the above example. This means a nologin shell will retain all the settings and configurations of the current shell, whereas a login with the hyphen will create a new shell with a clean login for the new user. It is a good practice to login using the hyphen.

sudo and root

The Linux operating system has implemented a very strict model for users. While the root user has the power to do anything and everything, a regular

user cannot make system changes directly. In the initial years of Linux, the solution to this challenge was to allow the regular user to switch to the root user through the su command until the required task was completed. This was seen as a disadvantage since a regular user could become a root user with all the privileges. This meant that they could make any changes to the system and even delete important directories. Also, switching to the root user via the su command meant that a regular user would then know the root user account's password. This was definitely not a practical solution.

This was when Linux introduced the concept of **sudo.** The sudo command simulates a regular user to be the root user and run specific root commands. The settings for sudo are defined in the file stored at /etc/sudoers. While the su command needed you to know the root user's password, the sudo command requires you to know only your password. This means the root user can assign certain privileges to regular users via sudo to perform system-related tasks without giving away the root password.

Let us look at the example below where a student user is running the usermod command through sudo access. The student user can use this access to modify another user account and lock that account.

[student@desktop ~]$ sudo usermod -L username

[sudo] password for student: studentpassword

Running commands through sudo has a huge benefit as well. All the commands that are run on a Linux system using sudo are logged in a file at /var/log/secure.

User Account Management

This section will teach you how you can create, modify, and delete user accounts that exist on a Linux operating system. You can manage local user accounts in Linux via numerous tools on the Linux command line. Let us go through them in brief.

useradd username

This command is used to add a new user to the system. The username that is passed along with the command is used to create the new user. The user gets created with the default parameters with the respective values in the file at

/etc/passwd when you do not use any other options with the command. Note that this command will not set a password for the new user and, the user will not be able to login into the Linux system until a password is set.

You can type the *useradd --help* command to get the manual for the useradd command that will give you a list of options that you can pass with this command. If you manually pass options for the useradd command, they will override the user's default properties in the /etc/passwd file.

Some parameters for a user, such as the UID and password aging policy, are picked up from the file at /etc/login.defs. This file is used only when a new user is created. Modifying this user does not affect the properties of the existing users on the system.

usermod username

This is a command that can be used to change existing users' properties in the Linux system. You can type the command ***usermod --help*** to get a list of all the options that can be used with this command. Let us see these options.

-c, --comment COMMENT	This option is used to add a value such as full name to the GECOS field
-g, --gid GROUP	The primary group of the user can be specified using this option
-G, --groups GROUPS	Associate one or more supplementary groups with user
-a, --append	The option is used with the -G option to add the user to all specified supplementary groups without removing the user from other groups
-d, --home HOME_DIR	The option allows you to modify a new home directory for the user
-m, --move-home	You can move the location of the user’s home directory to a new location by using the -d option

-s, --shell SHELL	The login shell of the user is changed using this option
-L, --lock	Lock a user account using this option
-U, --unlock	Unlock a user account using this option

userdel username

As the command suggests, this will delete the user account and its record from the /etc/passwd file; but this command does not delete the user's home directory. You can use the *userdel -r username* command to delete the user from the /etc/passwd file and delete their home directory too.

id

This command, as we discussed earlier, displays the details of the user that is currently logged in. Using another username along with the id command will show the basic details of the other user as well.

passwd username

This command can be used to set the initial password for a user that you create. It can also be used to modify the password for an existing user.

The root user can modify any user’s password and set it to any value. If the password strength criterion is not met, a warning will be shown, but the root user can simply retype the same password to set it for the user anyway.

If a regular user is trying to set a password, it will have to meet the criteria of having at least eight characters, it should not match the username, and it should not be a word that can be found in a dictionary.

UID Ranges

These are ranges that are reserved for specific purposes in a Linux operating system.

- UID 0 is always assigned to the root user.

- UID 1-200 is statically assigned by the system-to-system processes.
- UID 201-999 are assigned to the system process that does not own any file in
- the system. They are dynamically assigned whenever installed software requests for a process.
- UID 1000+ are assigned to regular users of the system.

Group Management

Like in the case of users, you can also create, modify, or delete groups on the system at a local level.

Remember that it is important for a group to exist before you can add users to that group. You can manage local groups through several groups in Linux. Let us go through these commands that are used for groups.

groupadd groupname

This command is used to add a new group. The groupname passed with the command is created on the system, and a GID is assigned to it. The group is defined in the file stored at /etc/login.defs.

You can also specify a particular GID by using the following option **-g GID**

[student@desktop ~]$ sudo groupadd -g 5000 ateam

You can also use the **-r** option along with this command to create a system-specific group, and the GID assigned to it will be from a range of system values for GID.

[student@desktop ~]$ sudo groupadd -r professors

Groupmod

This command can be used to modify the properties of a group that already exists in the system. You can use this command to perform tasks like changing the groupname's mapping to a GID. You can use the **-n** option with this command to change the name of the group.

[student@desktop ~]$ sudo groupmod -n professors lecturers

You can also pass the **-g** option with this command if you want to change the GID for an existing group.

[student@desktop ~]$ sudo groupmod -g 6000 ateam

groupdel

This command is used to delete a group from the Linux system.

[student@desktop ~]$ sudo groupdel ateam

Note that the groupdel command may not work on the primary group of the user. Just like in the case of userdel, you need to ensure that a user does not own any files after you delete a group.

Usermod

This is a user command but can be used to modify the group of a user. This can be done through the command *usermod -g groupname*

[student@desktop ~]$ sudo usermod -g student student

You can also use this command to add a user to a supplementary group through *usermod -aG groupname username*

[student@desktop ~]$ sudo usermod -aG wheel student

The **-a** option ensures that the modifications are being appended to the user's properties. If you do not use the -a option, the user will be added to the specified group and will be deleted from all other groups.

Password Management for User

This section will take you through how the password for a user on the Linux system is managed. We will cover the shadow password file used to password aging policies for a user and can also be used to manually lock a user account. In the beginning, Linux systems stored the encrypted password for a user in the file at the /etc/passwd, which was accessible to everyone. It was a secure path for a long time until attackers started targeting this file with dictionary attacks. This was when Linux developers decided to move the password to a more secure location at the /etc/shadow file. This new file has

features to set password expiry through password aging policies.

The password today consists of three parts. Let us look at a password hash, for example.

1gCLa2/Z$6Pu0EKAzfCjxjv2hoLOB/

1. The **1** in this hash tells you the hashing algorithm that is used. The number 1 refers to the MD5 hashing algorithm. If a SHA-512 hashing algorithm were used, you would see the number 6 instead.

2. The part where you see **gCLa2/Z** indicates the type of salt used for encrypting the hash. Initially, it is a randomly chosen salt: the unencrypted password and the salt in combination form the encrypted hash. Using a salt ensures that even if two or more users end up having the same passwords, their hash entries will not be the same in the /etc/shadow file.

3. The **6Pu0EKAzfCjxjv2hoLOB/** is the encrypted hash.

When a user logs in to the Linux system, the system checks if the user's credentials exist in the /etc/shadow file. The system then takes the user's password and validates it against the password hash stored for that user in the /etc/shadow file. If it matches, the user is logged in. If it does not match, the user receives an error stating that the password is incorrect.

Let us go through the format of the /etc/shadow file.

- name:
- password:
- lastchange:
- minage:
- maxage:
- warning:
- inactive:

- expire:
- blank:

name*:* This is the username of a user on the Linux system that a user enters to login.

password*:* This field is where the password for the user is stored in an encrypted format. An exclamation mark at the beginning of this field would mean that the password is locked.

lastchange*:* This field maintains the timestamp when the last change occurred for the user's password.

minage*:* This field defines the minimum number of days before a password needs to be changed. 0 would indicate that there is no minimum age set for the account.

maxage*:* This field defines the maximum number of days before a password needs to be changed.

warning*:* This field shows the warning period when a user would start seeing password expiry notifications. Again, if this is set to 0, the user will not receive any warnings.

inactive*:* This field shows the number of days after password expiry, the account will stay inactive. The user can still log in to the system during this period to reset their password. If the user fails to do so in the number of days defined in this field, the user account will be locked and become inactive.

expire*:* This field shows the date when the account is set to expire.

blank*:* This field is a blank field and is reserved for future use.

Password Aging

The password aging policy in Linux is in place to ensure the security of a user account. By default, the policy keeps a 90 days interval after which a user is forced to change their password. This policy's advantage is that even if someone gains access to a user's password, they will be cut off from it once the password hits 90 days. This policy also has a drawback that users may

tend to write their password down somewhere when they keep changing it and can’t memorize it.

Password aging can be enforced in two ways in a Linux operating system.

1. By using the **chage** command on the command line.
2. Through the graphical interface in the User Management settings.

If you have sudo privileges, you can use the chage command with the **-M** option to set the user's password validity. Look at the example below.

[student@desktop ~]$ sudo chage -M 90 alice

In the above example, the password validity for the user alice is being set to 90 days. After 90 days, Alice will be forced to reset her password. You can simply disable password aging by setting the number of days to 9999, which is equal to 273 years.

Access Restriction

Using the chage command, you can set the expiry for a user account. Once the expiry is reached, the user will not be allowed to log in to the system. You can also use the **usermod** command with the **-L** option to lock a user account.

[student@desktop ~]$ sudo usermod -L alice

[student@desktop ~]$ su - alice

Password: alice

su: Authentication failure

If a particular user is no longer needed on a system, you can use the usermod command to simultaneously lock and expire the account.

[student@desktop ~]$ sudo usermod -L -e 1 alice

A user will not be able to login to the Linux system when their account is locked. This practice is mostly followed by organizations to lock the user

accounts of employees who have left the organization. The account can still be unlocked later if needed by using the **usermod -u** *username* command in the event that the same employee rejoins the organization.

The Nologin Shell

The nologin shell comes into the picture when you want to give login access to a user but do not want to give them a shell to interact with the system. The simplest example of this is a mail system that needs the user to have a username and password to check their mail, but the user does not need to access the complete Linux operating system. A user can be given a nologin shell at the time of user creation, or their shell can be modified later. This is done by defining the user's shell as **/sbin/nologin**. If this is done, a user will not be able to log in to the Linux system.

[root@desktop ~]# usermod - s /sbin/nologin student

[root@desktop ~]# su - student

Last login: Tue Mar 5 20:40:34 GMT 2015 on pts/0

The account is currently not available.

When you assign a nologin shell to a user, they cannot interact with the system, but at the same time, they also have some access to the system. The user will be able to use web applications for file transfer, etc.

Chapter Nine

Linux File System and Permissions

This chapter will teach you about the Linux file system's permissions and how you can change file and directory permissions using command-line tools. By the end of this chapter, you will know how permissions in the Linux operating system work and how they can be applied to files and directories to access and restrict access.

Linux File System Permissions

File permissions are a feature of Linux through which you can control access to files and directories. The file system model of Linux is very simple, and it has a flexible nature such that a new Linux user can easily understand it to apply permissions to files and directories.

There are three types of users on a Linux system. The permissions are applied to these three users with respect to a file or directory. The three types of users are as follows.

1. user
2. group
3. other

The hierarchy of permissions is such that a user can override group permissions, and a group can override other permissions.

There are three categories of permissions that apply to a file or directory in Linux.

1. read
2. write
3. execute

Let us understand the effect of these permissions on a file and directory.

Permission	Effect on Files	Effect on Directories
r (read)	Read access to file content	Contents of a directory that is filenames will be listed.
w (write)	Write access to file content	Contents of the directory where files can be created or deleted.
x (execute)	The file can be executed as a command	Contents of the directory where files can be accessed subject to the permission of the file itself.

By default, a user will have access to read and execute a Linux system file if it is an executable file. If a user has only read access, they can read the contents of the file but will not be able to see any other information, such as the permissions set on the file or the timestamp of the file. If a user has only 'execute' access to a file, they will not be able to list down the file in a directory but can execute the file if they already know the file's name.

If the user has write access to a directory, they can delete any file in the directory even if they do not have any permission on the actual file itself.

Let us see how you can know the permissions that are assigned to a file on the Linux system.

You can use the **ls** command with the **-l** option to list down the contents of a directory along with the details for its ownership and permissions.

[student@desktop ~]$ ls -l test

-rw-rw-r--. 1 student student 0 Feb 5 15:45 test

If you want to know the permissions and ownership of the directory itself and its contents, you can use the **-d** option.

[student@desktop ~]$ ls -ld /home

drwxr-xr-x. 5 root root 4096 Feb 8 17:45 /home

The read permissions in Linux is equivalent to the List Folder Contents permission that is present in Windows.

The write permission in Linux is equivalent to the Modify permission in Windows.

The Full Control option in Windows is equivalent to the root user's permission in the Linux file system.

Using the command line to manage permissions

This section will teach you how to manage permissions and ownership for a file and directory in Linux using the command line.

Changing Permissions of Files and Directories

You can use the command **chmod** in Linux to change the permissions of files and directories. chmod is short for change mode because the permissions are also known as the file or directory mode in Linux. The syntax of this file is pretty simple. The command is followed by the permissions you want to set on the file or directory, followed by the filename or the directory name. There are two ways to provide these instructions, numerically and symbolically.

Let us go through the symbolic method first, where the syntax is as follows.

chmod WhoWhatWhich files directory

- Who is the user u, group g, other o, and a for all
- What is + to add, - to remove, = to set exactly
- Which is r for read, w for write, and x for execute

You will need to use the letter to specify the various groups you want the permissions to change for. As shown above, u stands for user, g for group, o for others, and for all.

When you are using the symbolic method to set permissions, it is not necessary to specify a new set of permissions for a file or directory. You can simply modify the existing permissions. This can be done by using the symbols, +, -, and = to add, remove, or replace permissions, respectively.

Permissions are defined using three letters where r is for read, w for write, and x for execute. If you are using the symbolic method to define permissions for a file and use a capital X, the execute permission will be applied to a file only if it is a directory or the file already has the execute permission set for the user, group, and others.

The syntax for the numeric method looks as follows.

chmod ### files|directory

- Every position of the # represents an access level viz. User, group, and other
- # is the sum of read r=4, write w=2, execute x=1

The numeric method can be used to set permissions on a file using three digits and sometimes the 4th digit for special permissions. Every digit for permissions can be a number between 0-7, and it shows the combination of possibilities that we can have with read, write, and execute.

If we understand the mapping between numeric and symbolic values, we can also learn to make the conversions easily. Every digit in the numeric method stands for permissions set for a particular group. Starting from left to right, the first digit represents permissions for a user, the second digit for a group, and the third digit for others. For each of these groups, read, write and execute permissions can be set by using a combination of 4, 2, and 1, respectively.

The symbolic representation of permissions looks like **-rwxr-x---**

In the example for **-rwxr-x---** the user has a permission of **rwx** that stands for read, write and execute. If we were to convert this to a numeric form, it would be 4, 2, and 1, and the sum of it is 7.

Moving on to the permissions for the group, it is **r-x**. So the group has ‘read and execute’ permissions but no write permissions. Converting this to the numeric form, it would mean that the group has 4 and 1 permissions, and the sum of it is 5.

Finally, the permissions for others are set as **---,** which means others do not

have read, write or execute permissions, and the numeric representation of this would be 0.

Combining the permissions for all three types of users for this file, the permissions as a whole for this file would be **750**.

A converse operation can also be done where we have the numeric permissions for a file and convert it to the symbolic form.

Consider the permissions **640**.

We know that the digit on the extreme left is for the user permissions. The number 6 can be dissected to be a sum of 4 and 2. That means the user has read and write permissions and no execute permission that can be shown symbolically as **rw-**

Moving to the digit for the group, it is 4. This means that the group has only read permission and no write and execute permissions. This can be shown symbolically as **r--**

Finally, the digit for others is 0. This means others do not have read, write, or execute permissions. This can be shown symbolically as **--**

As a whole, the permissions for this file will be represented symbolically as **rw-r-----**

Note that you can use the **-R** option with the chmod command if you want to apply permissions recursively to files under a directory tree.

Changing the Ownership for Files and Directories

The default ownership for a newly created file in Linux lies with the user who created the file. The default group owner for that file is also the primary group of the user who created the file. Access to files and directories can be controlled by changing the user and group ownership for them.

You can use the **chown** command to change the ownership of a file or directory in Linux. Let us look at an example.

[root@desktop ~]# chown student newfile

The above example shows that the ownership of the file newfile is being

changed to student.

The **-R** option can be used with the chown command to change the ownership recursively for all files under a directory. You can use the command as follows.

[root@desktop ~]# chown -R student parentdirectory

You can use the chown command to change the group ownership for a file and directory as well. The command is followed by the group name prefixed with a colon :

Have a look at the example below.

[root@desktop ~]# chown student:admins newfile

This command will change the group ownership of the file to admins.

Do note that only a root user has the right to change ownership for a file or directory; but if you are the owner of the file or directory, you can change its ownership. A non-root user can change a file's ownership to a group that they are already a part of.

Managing File Access and Default Permissions

This section is about special permissions on a file or directory in Linux. This can be a little overwhelming as a beginner, but we have included it since you have already learned about permissions. We will create a directory, and files under this directory will have write access by default on them for users of the group that owns the directory. This is achieved through special permissions known as sticky bits.

Let us see what special permissions are and how you can apply them. There is a bit known as the **setuid** and **setgid** on the permissions. This allows an executable file to run as the user of that file or the group of that file and not as the user that ran the actual command.

An example is the **passwd** file. If you look at the permissions of the passwd file, they look like this.

[student@desktop ~]$ ls -l /usr/bin/passwd

*-rw**s**r-xr-x. 1 root root 34598 Jul 15 2011 /usr/bin/passwd*

When there is a sticky bit set on a file, it restricts the file from being deleted. Only the user owner of that file or the root user can delete such files. An example of this is the /tmp directory.

[student@desktop ~]$ ls -ld /tmp

*drwxrwxrw**t** 39 root root 4096 Jul 10 2011 /tmp*

The setgid bit lets a file inherit permissions from its parent directory rather than the user setting its permissions.

Let us go through how special permissions affect files and directories.

Special Permission	**Effect on Files**	**Effect on Directories**
u+s suid	The file will execute as the user who owns the file and not the user who ran the command	No effect
g+s sgid	File executes as the group that owns the file	The group owner of the newly created file in the directory will match the group owner of the directory
o+t sticky	No effect	Users who have write access to the directory can only delete files owned by them. They cannot delete or force writes to files owned by other users

Let us see how we can set special permissions to files and directories.

- Symbolically, setuid is **u+s,** setgid is **g+s,** and sticky is **o+t**

- Numerically, the special permissions use the fourth bit that precedes every user's first digit. setuid is 4, setgid is 2, and sticky is 1.

Let us try to understand default permissions. A process that creates a file sets the default permissions for that file. For example, if you created a file using a text editor such as vim or nano, the file will have read and write permissions for everyone by default. It will not have the execute permission for everyone, though. Files created by compilers are binary executable by default. Those files will have executable permissions.

The **mkdir** command can be used to create directories, and these directories have read, write and execute permissions by default.

As per research, permissions are not set on a file or directly as soon as they are created. The **umask** of the shell process is known to clear these permissions. You can use the umask command without any options or arguments to see the shell's current umask value.

[student@desktop ~]$ umask

0002

Every process in the Linux system has a umask associated with it. The umask is basically an octal bitmask that clears the permissions of newly created files and directories that are created by a process. If there is a bit set for umask, the corresponding permission gets cleared on a newly created file.

Let us look at an example.

In the above example, the bitmask value is 0002. This means that the bit for other users is set to 2. By this, we can infer that the special user, and group permissions will not be cleared since they are set to 0. Permissions for others will be cleared since the umask bit is set to 2.

The default values of umask in the system for users of the bash shell are defined at /etc/profile and /etc/bashrc files.

Chapter Ten

The Linux Boot

Have you ever wondered what exactly happens when you switch on your Windows or Linux based computer? This chapter will take you through the boot process of the Linux system. This will give a deep understanding of what exactly happens from the time you switch on your Linux computer till the time you are presented with the login screen and, eventually, the desktop.

Understanding the Linux boot process is important if you want to get good at playing around with the Linux configuration. This knowledge will also teach you to troubleshoot any issues when your computer does not start as expected. You will learn about the Linux boot and startup sequence and about the GRUB2 bootloader. You will also learn about certain system daemons like the systemd initialization.

There are two parts to initialize a Linux operating system and make it ready to use for a Boot and Startup user. The boot process is initiated when you power on the computer and concludes when the control is taken over by the kernel and passed to systemd. Once systemd takes over, the startup process is initiated and concluded when the Linux system stabilizes and reaches a state where a user can start using it.

It is easy to understand the Linux boot and startup process. It consists of the following stages.

- BIOS POST
- Boot loader (GRUB2)
- Kernel initialization
- Initialize systemd

We will be discussing the Linux boot and startup process in detail in this

chapter using the GRUB2 bootloader, as it is used in a majority of Linux distributions. A few Linux distributions still use old software for the boot and startup process, but we will not be going through them.

The Boot Sequence

The boot sequence can be initiated in a couple of ways for Linux. The simplest one is turning on the power for your Linux system. If there is a user created on the system or logged in as a root user, you can initiate a reboot of the system via the command line to trigger the boot sequence.

BIOS POST

The first step of the boot sequence in Linux is not related to Linux and its operations. The first step is all about initiating the hardware on the system and is common across all operating systems. When you press the power button for your system, a test known as the Power On Self Test or POST is short is initiated. POST is a part of the Basic Input Output System or BIOS.

BIOS was introduced for the first time in 1981 when IBM developed its first computer. POST is a module of BIOS that is in place to check if all the hardware on the system is functional. IF POST fails, it indicates that there is something wrong with the system. The system will be in an unusable state, and BIOS would not be initiated.

POST and BIOS's function is to check if all the hardware needed to start the system is functional. They also follow up with a BIOS interrupt known as INT 13H, which is responsible for finding bootable devices such as a hard disk and initiating it by locating its boot sectors. When the interrupt locates the first boot sector containing a valid boot record, it is loaded into the memory and the control gets transferred to the code present in the boot sector.

You can say that the boot sector is the first stage of the boot loader. There have been around three bootloaders that have been used across Linux distributions over the years, LILO, GRUB, and GRUB2. The latest and most popular bootloader is GRUB2, and every latest Linux distribution now uses GRUB2.

GRUB2

GRUB2 stands for GRand Unified Bootloader version 2. GRUB2 is a module that locates the kernel of the operating system and loads it into system memory.

Linux has never officially declared any stages for GRUB2, but we'd like to tell you that GRUB2 operates in 3 stages. Let us go through them.

Stage 1

As already mentioned in the POST and BIOS section, when the POST process is successful, the control is taken over by BIOS, and BIOS starts looking up any bootable devices. It then locates the Master Boot Record (MBR) on the bootable device and then loads it into system memory from where the boot records are then executed. The bootstrap code for GRUB2's stage 1 is very small because the requirement is that it fits within the first 512-byte sector on the bootable device along with the metadata of the partition table. Owing to this, the space allocated to the bootstrap code is only 446 bytes. This 446-byte file is known as boot.img, which does not contain the partition table.

The boot record, owing to its small size, does not maintain any information about the file system. The main objective of the GRUB2 bootloader stage 1 is to locate the next stage. Between the boot record and the bootable device's first partition lies stage 1.5 of the GRUB2 bootloader. Stage 1 loads stage 1.5 into memory, and control is passed to stage 1.5.

Stage 1.5

Stage 1.5 will always lie between the boot record and the first partition of the bootable device. This space was not allocated to anything for a lot of years because of technical issues. The first partition on a hard disk starts at sector 63. The master boot record starts at sector 0. This means that between the master boot record and the first partition of the hard disk drive, there are 62 sectors, or 512-byte sectors, or 31,744 bytes. This is the space where the core.img file can be stored that contains stage 1.5 of GRUB2. The size of the core.img file is 25,389 bytes. This means it can be placed comfortably after the master boot record and before the first partition.

Since there is sufficient space to store the code for GRUB2 stage 1.5, which does not need a lot of space, the additional space is used to contain file system drivers such as FAT, EXT, and NTFS. This means that a standard EXT file system is enough to store the code for GRUB2 stage 2. The location for the GRUB2 stage 2 code is at /boot/grub2.

Stage 2

All the files that are needed to execute stage 2 of GRUB2 are located at /boot/grub2 and its subdirectories. Stage 2 of GRUB2 does not have any image files like it did in stage 1 and 1.5. Instead, it loads runtime modules when needed from the directory at /boot/grub2/i386-pc.

The main objective of GRUB2 stage 2 is to find the Linux kernel and load it into memory so that the control can be passed to the kernel. The Linux kernel files are stored in the /boot directory. One can identify Linux kernel files easily since they are all prefixed with **vmlinuz**. You can list down all the kernel files for your Linux system by simply listing down the /boot directory contents.

So when your system boots up, you get a screen where all the kernels available on your Linux system are listed, and you can select one of them to pass the control to the respective kernel.

Kernel

All the kernels available today are stored in a compressed format for space optimization, but they can self-extract themselves. We know that the Linux kernel can be located in the /boot directory. When you select a particular kernel on the boot menu, it gets executed and starts extracting itself so that the system can use the files. When the kernel is executed, it makes a call to **systemd** and eventually passes the control to systemd.

This is the end of the Linux boot process. At this point in time, both the bootloader and systemd are in a running state but are not useful to the end-user since nothing else is running yet.

The Startup Sequence

The Linux startup sequence is initiated after the boot sequence concludes.

The startup sequence is responsible for bringing the Linux operating system to a usable state for an end-user so that they can log in and start using the system for their tasks.

Systemd

Systemd is called the mother of all processes in the Linux operating system. It is responsible for initiating the Linux operating system and bring it to a usable state. Before the systemd service was developed, the same task was performed by the **init** service, but systemd has many more functions compared to init, such as initiating and managing system service, mounting file systems, etc.

The first task for systemd is to mount all the file systems that are defined in the configuration file located at /etc/fstab. systemd also has access to all the other configuration files stored in /etc. systemd brings the system into a state or target as defined in the file stored at /etc/systemd/system/default.target. If it is a desktop, the default target is usually the **graphical.target**. If it is a Linux server, the default target is usually the **multi-user.target**.

Targets and services are units of the systemd service. The configuration file for a target defines the dependencies of that particular target. These dependencies are initiated by systemd. The dependencies refer to services needed to run the Linux operating system based on the initialized target. When systemd loads all the target's dependencies, it can be said that the system is up and running or that the system is running at that particular target level.

The startup sequence has two checkpoints, **sysinit.target** and **basic.target**. It is a fact that systemd starts services simultaneously, but some services need to be initiated before other services. A checkpoint is passed only when all the services and targets for that particular checkpoint have been initiated successfully.

The sysinit.target checkpoint is reached only after all its services and targets have been initiated. The sysinit.target has multiple dependencies, and these are initiated parallelly within sysinit.target.

The sysinit.target first starts all the low-level services that are needed to reach

the basic.target. The basic.target then initiates some additional services to reach the next target.

Once both these checkpoints have been reached successfully, the system can move to the **graphical.target** or **multi-user.target**. But note that the Linux system can reach the multi-user.target even before the dependencies of the graphical.target have been initiated.

You will see a text-based login prompt if the default target is set to multi-user.target. This target is commonly used by Linux servers that have Linux installed on them without any graphical interface. If you have installed Linux on a personal desktop and have included the GNOME desktop along with the installation, the default target will be the graphical.target. In such a case, you will be presented with a graphical interface to login into the Linux desktop.

This is where the startup sequence concludes. You can simply enter your username and password on the text-based login prompt or the graphical interface login screen to start using your Linux system.

Every modern-day Linux operating system functions with GRUB2 and systemd as its core components. Both these components work together to load the Linux kernel, start the services, and make the Linux system usable for any end-user.

Chapter Eleven

Introduction to Shell Scripting

The open-source programming language developed for Linux is known as Shell Scripting. Simply put, shell scripting is a series of individual Linux commands put together to be executed automatically. A shell script may have a single line of code or multiple lines of code that are stored in a single file such that a user does not need to type them manually every time they need to perform a specific task.

We will introduce you to the basics of shell scripting in this chapter, and we will also briefly discuss some advanced concepts of shell scripting. This chapter will serve as a launchpad for you into the world of shell scripting.

Writing a Shell Script

You can use a text editor to write a shell script. Linux has a set of inbuilt text editors such as nano, vi, and vim. You can use any of these to start writing a shell script. One thing you need to ensure is that the files you create for your shell script are executable. You can make any file executable by running the following command on the filename.

$ chmod a+x filename.sh

And then, you just need to store the file in a location that is easily accessible by the Linux operating system.

The following steps go into the creation of a shell script.

1. Create a new file using the text editor of your choice. Save the script file with a .sh extension.
2. The first line of your script file should always be **#! /bin/sh**
3. Type in some Linux commands.

4. Save the file as filename.sh
5. Use the following command on the Linux shell to execute your script.

bash filename.sh

#! is known as a shebang operator. It tells the script the location of the interpreter of the Linux system. When you type the first line in your shell script as #! /bin/sh the Linux operating system knows that the interpreter to execute the script is located at /bin/sh

Let us create a simple shell script.

```
#!/bin/sh

ls
```

Save this script as list.sh

You can run this script using the following command on the shell.

```
[student@desktop ~]$ bash list.sh

Documents    Movies        Books        Pictures
```

The result of this script is the **ls** function that lists down the contents of a directory.

Commenting Your Script

Commenting is used in every programming language, and shell scripting is no exception to it. You can use the following syntax to add comments to your shell script.

```
#this is a comment
```

If we take the above script as an example, you can add a comment to it as follows.

```
#!/bin/sh
#this is my first script
```

ls

You can also add comments as multiple lines to your shell script. You can do so by placing your comments between **:'** and. **'**

Consider the example below.

#!/bin/bash

: '

This script calculates

the square of 5.

'

*((area=5*5))*

echo $area

So the string between :' and ' that is this script calculates the square of 5 is commented out from the script.

Shell Variables

Variables in any programming language are used to store data in the form of characters and numbers. Shell scripts also support variables that store data that can be accessed by the Linux shell.

For instance, the simple example below shows a variable that stores some data and then prints it.

variable ="Hello"

echo $variable

Let us look at another example of a shell script that uses variables.

#!/bin/sh

echo "what is your name?"

read name

echo "How do you do, $name?"

read remark

echo "I am $remark too!"

How Does This Script Work?

When you run this script, the system will print the question, “what is your name?”

You will then see that the system is expecting input from you. If you type Mark, the string Mark is then stored in the variable **name**. In the next line of code, the script uses the name provided to ask the next question, “How do you do Mark?” where the variable **name** gets replaced by the string entered by you earlier. The system again expects the answer to this question as input from you. When you type it, it is again stored in another variable called **remark**. So if your answer is “fine,” the script will replace the variable **remark** with your string in the next line of code and print the output as “I am fine too!”

More Shell Scripting Examples

Hello World

The first program everyone types to learn a new programming language is often Hello World. It is a simple program that will print the output as “Hello World.” Use a text editor like nano or vim and create a file named helloworld.sh and type the following lines of code in it.

#!/bin/bash

echo "Hello World"

Save the file and change the permission for the file to make it executable.

$ chmod a+x hello-world.sh

Now run one of the following commands to run your script.

$ bash hello-world.sh

OR

$ ./hello-world.sh

This will print the string that you have passed with the echo command, which in this case is “Hello World.”

Echo Command

You can use the echo command to print outputs to the bash shell. If you have ever read about C programming, it is equivalent to the function of the command **printf** in C programming.

Create a new shell script name echo.sh and type the following lines of code in it.

```
#!/bin/bash

echo "Printing text"

echo -n "Printing text without newline"

echo -e "\nRemoving \t special \t characters\n"
```

Give executable permissions to the script and run it to see the output. The **-e** option passed with the echo command tells the command that the string that is being passed contains special characters and needs special execution.

The While Loop

If you want to run a command multiple times in your script, you can either type it multiple times or place it inside a while loop.

Create a new script called while.sh and place the following lines of code in it.

```
#!/bin/bash

i=0


while [ $i -le 2 ]

do
```

```
echo Number: $i
((i++))
done
```

The flow of the above while loop is as follows.

```
while [ condition ]
do
commands 1
commands n
done
```

Run the script now to see how it works exactly.

The For Loop

The For loop is similar to the while loop and lets you run a line of code or multiple lines of code in an iteration. Let us look at an example.

```
#!/bin/bash
for (( counter=1; counter<=10; counter++ ))
do
echo -n "$counter "
done
printf "\n"
```

Save this script as for.sh and run it to see what happens. It will print numbers from 1 to 10 on the screen. Feel free to make changes to the script to try and print another set of numbers.

The If Statement

The If statements are commonly occurring statements in the Linux shell

script. They are used to create conditions and have the following flow.

```
if CONDITION
then
STATEMENTS
fi
```

If the given condition is true, the statement gets executed. The **fi** keyword marks the end of an If statement. Let us look at an example.

```
#!/bin/bash
echo -n "Enter a number: "
read num
if [[ $num -gt 10 ]]
then
echo "Number is greater than 10."
fi
```

This code will ask the user to enter a number. Now the If condition says that if the number is greater than 11, then execute the next line of code. If the number is less than 10, the program will not print any output at all.

-gt stands for greater than, **-le** stands for less than, and **-le** and **-ge** stand for less than equal to, and greater than equal to respectively.

The If-Else statement

You can add more control to your If statement by adding Else to it. The logic is simple. When the If statement is true, the instructions under it are executed; else, the instructions under the Else statement are executed.

Let us look at an example.

```
#!/bin/bash
```

```
read n

if [ $n -lt 10 ];

then

echo "It is a one-digit number"

else

echo "It is a two-digit number"

fi
```

This is a similar code as above where if the number provided in the input is less than 10, the program will print “It is a one-digit number,” else it will print “It is a two-digit number”

The AND Operator

If you want to add multiple conditions that need to be satisfied before the next line of code is executed, you can use the AND operator. All the conditions that are separated by an AND operator must be met. If not, the code after the AND operator will not be executed. Let us have a look at an example script to understand how it works.

```
#!/bin/bash

echo -n "Enter Number:"

read num

if [[ ( $num -lt 10 ) && ( $num%2 -eq 0 ) ]]; then

echo "Even Number"

else

echo "Odd Number"

fi
```

The AND operator is represented by the **&&** sign.

Now, as per this script, if the number input by a user is less than 10 and if the remainder for it is 0 when it's divided by 2, the program will print "Even Number." If not, it will print "Odd number."

The OR Operator

If you want to add multiple conditions and need only one of them to be satisfied before the next line of code is executed, you can use the OR operator. Any one of the conditions that are separated by an OR operator must be met. If not, the code after the OR operator will not be executed. Let us have a look at an example script to understand how it works.

```
#!/bin/bash

echo -n "Enter any number:"

read n

if [[ ( $n -eq 15 || $n -eq 45 ) ]]

then

echo "You won"

else

echo "You lost!"

fi
```

The || sign represents the OR operator.

As per the above script, if the user's input is equal to 15 OR if it is equal to 45, the program will print "You won." Otherwise, it will print "You lost!"

The Elif statement

The elif statement is an extension of the if-else statement. Let us see how you can add more conditions to your script using the elif statement.

```
#!/bin/bash

echo -n "Enter a number: "
```

```
read num
if [[ $num -gt 10 ]]
then
echo "Number is greater than 10."
elif [[ $num -eq 10 ]]
then
echo "Number is equal to 10."
else
echo "Number is less than 10."
fi
```

The above script is self-explanatory. You can change some portions of it to try new things as well.

The Switch Construct

Another powerful tool available in Linux bash scripts is the switch construct. If you want to create nested conditions, the switch construct is very convenient. It is an easier alternative to the if-else-elif nests. Have a look at the example below.

```
#!/bin/bash
echo -n "Enter a number: "
read num
case $num in
100)
echo "Hundred!!" ;;
200)
echo "Double Hundred!!" ;;
```

```
*)
echo "Neither 100 nor 200";;
esac
```

The conditions for a switch are places between the case and esac keywords. The condition is then placed before the **)** symbol.

Concatenation

Concatenation is a function of the wider term that is known as string processing. String processing is playing with strings such that you can get the desired output. Concatenation is a string processing function that appends two or more strings together. Let's have a look at an example.

```
#!/bin/bash
string1="Ubuntu"
string2="Linux"
string=$string1$string2
echo "$string is a great operating system for Linux beginners."
```

The output of the above script will be "UbuntuLinux is a great operating system for Linux beginners "

Adding Two Numbers

Linux scripts are very easy to perform arithmetic operations. The script given below takes two numbers as inputs from the user, adds them, and prints the sum.

```
#!/bin/bash
echo -n "Enter first number:"
read x
echo -n "Enter second number:"
```

```
read y

(( sum=x+y ))

echo "The result of addition=$sum"
```

Adding numbers is very straightforward using a bash script.

Adding Multiple Numbers

Adding multiple numbers can be a bit tricky, and you will have to use loops. Let us look at an example.

```
#!/bin/bash

sum=0

for (( counter=1; counter<5; counter++ ))

do

echo -n "Enter Your Number:"

read n

(( sum+=n ))

#echo -n "$counter "

done

printf "\n"

echo "Result is: $sum"
```

Run this script and see how many numbers this script allows you to add before printing the sum.

Linux shell scripts can be used for any task that you wish to perform on the Linux operating system. It has unlimited potential, and there is literally no limit to what you can achieve with it. We would encourage you to practice every bash script that we have provided in this chapter and even make your own changes to it to see how the output changes.

Conclusion

Linux operating systems and their distributions are very secure and therefore preferred by individuals and businesses all over the world today. With respect to personal users, Linux has distributions such as Ubuntu that help a user seamlessly transition from any other operating system to Linux. Linux has proved to be very beneficial for students since it is free to install and is open-source in nature.

With minimal monetary investment, a user can have a functional operating system and use it for all their day-to-day tasks. The various distributions of Linux suit the needs of almost every user in the world. With respect to enterprises and businesses, user data is the most important entity, and data losses can affect the brand image of a business on a large scale.

Over the years, Linux has proved to be a secure and stable operating system for businesses to run their applications on and store critical customer data. Security patches for Linux operating systems are also rolled faster than any other operating system today. This has made Linux the ideal operating system for any business that has security as its major concern. We have also gone through multiple chapters in this book that show us how many features offered by Linux are unparalleled compared to other operating systems.

We have discussed the Ubuntu distribution of Linux in this book, and we hope that this distribution has made you enthusiastic about your journey into the world of Linux. Ubuntu is just the beginning of Linux for you, and there is so much more to explore in the domain of computer science by choosing Linux as your preferred operating system. Once you have become comfortable as a desktop-user of Linux, we would encourage you to explore other Linux distributions such as Kali Linux, Red Hat Enterprise Linux, etc.

Red Hat Enterprise Linux would help you get into a profile known as Linux system administration. Businesses that use Linux for their server always require someone to administer the server such that the servers are always up and running to ensure that their business is always running on the internet.

Kali Linux is used by businesses that have penetration testing teams to test

their servers for any security vulnerabilities. So this distribution comes into the picture with respect to cybersecurity.

Other than these two distributions, there are many other flavors of Linux waiting for you, and the potential to use Linux is unlimited.

References

10 Things To Do After Installing Ubuntu 20.04 LTS. (2020, April 23). OMG! Ubuntu! https://www.omgubuntu.co.uk/2020/04/things-to-do-after-installing-ubuntu

An introduction to the Linux boot and startup processes. (n.d.). Opensource.com. https://opensource.com/article/17/2/linux-boot-and-startup

Chawla, V. (2020, October 8). Windows Vs macOS Vs Linux: Best OS For Cybersecurity. Analytics India Magazine. https://analyticsindiamag.com/windows-vs-macos-vs-linux-for-cybersecurity/

How to Install Ubuntu 20.04 LTS {With Screenshots}. (2020, May 25). Knowledge Base by PhoenixNAP. https://phoenixnap.com/kb/install-ubuntu-20-04

Kiarie, J. (n.d.). 10 Linux Distributions and Their Targeted Users. Www.tecmint.com. https://www.tecmint.com/linux-distro-for-power-users/

Linux History - javatpoint. (n.d.). Www.javatpoint.com. https://www.javatpoint.com/linux-history

Prakash, A. (n.d.). 16 Things to do After Installing Ubuntu 20.04 - It's FOSS. Https://Itsfoss.com/. https://itsfoss.com/things-to-do-after-installing-ubuntu-20-04/

Shell Scripting Tutorial for Linux/Unix Beginners. (2019, November 21). Guru99.com. https://www.guru99.com/introduction-to-shell-scripting.html

What is Linux? - Linux.com. (2018). Linux.com. https://www.linux.com/what-is-linux/

What's new in Ubuntu Desktop 20.04 LTS? (n.d.). Ubuntu.

https://ubuntu.com/blog/whats-new-in-ubuntu-desktop-20-04-lts

LINUX
LINUX COMMAND LINES AND SHELL SCRIPTING

Andy Vickler

Introduction

This book on the Linux command line and shell scripting contains proven steps and strategies for using the keyboard to type commands and writing shell scripts to create your customized programs. Linux is tougher than Windows and Mac operating systems, and it is the securest of the three. When you have learned Linux, you can create your custom software and use it to do common daily tasks. You can customize how you want your computer should work.

Linux is fun to operate, but it is not easy to operate. Unlike Windows that has a graphical interface, Linux has a command line. If you are unfamiliar with what a command line is, you should refer to Dos or Command Prompt in Windows. It is a black screen on which you have to enter commands and complete your daily tasks.

I have written this book in two-parts: the first part deals with commands and the second part deals with shell scripting. Unless you have a good grasp of Linux commands and shell scripting, you cannot expertly operate the system. I have packed the book with the most used and important commands that you'll need. I have also included a few results you can see on the command line after entering a command.

The shell scripting section can be tough. If you have a programming background, it is fun to read. To practice scripting, you will have to work in an editor, but don't worry, you will find a chapter on Linux editors later in the book. You can write your own scripts or copy mine and paste them into the editor to practice. For convenience, I have included the results of each script so that you may compare them. I have also included lots of basic script examples on how to use commands.

Chapter One

What is Linux?

As a Linux student, the first thing you should learn is Linux's origins to understand the concept behind this operating system. Linux is a software layer between the hardware and the software in a computer operating system. It allows you to do productive things and create custom programs on your computer. In simple words, an operating system is a medium between the software and a computer system's hardware. An operating system allows you to store data on your storage devices like hard drives, solid-state drives, and USBs. It manages the transmission of data from one element to another; for example, it oversees data flow from the operating system to printers in your office or home. If you have installed a normal Windows environment such as the Microsoft Windows operating system on your computer system, the Windows operating system runs the hardware. It controls the mouse, keyboard, printers, scanners, and other accessories. You will have to install Microsoft Office, Adobe readers, pdf converters, and other software as per your needs. You will pay for each program and have them installed on your system.

Linux is the same as the Windows operating system regarding the process of controlling the hardware. It is different because it acts as a medium between the instructional code of the software and the physical device. The biggest difference is that of the software you will use in the Linux operating systems. The software will be of a different type as compared to the ones that run on Windows systems. You cannot install and run Microsoft Office or Adobe Photoshop on Linux environment. Linux runs different servers like web virtualization servers, Apache servers, database servers, etc.

However, Linux has several distributions that are made for personal desktop computers. These distributions are similar to macOS and Windows operating systems. They run the same type of programs like word processors, image editors, and games. These Linux distributions appear to be more targeted for

the home users searching for a free system alternative.

Linux did not kick-off as an operating system or a challenger to the Windows operating system. In the start, Linux happened to be a kernel that was created by Linus Torvalds. Linus was a student then at the University of Helsinki. The kernel is still useful in the system. In the start, the Linux kernel was used along with the GNU operating system. You can say that the GNU system was incomplete without the kernel.

A kernel is defined as an integral component of Linux. A kernel is considered the central part of operating systems, responsible for all the interfacing of applications and hardware. There are two types of kernels in the market now, namely Unix-like kernels and Windows kernels.

The Birth of the Linux Operating System

Between 1991 and 1994, Linus took a step further to create the Linux operating system. He combined the GNU OS with Linux Kernel. At the start, he wanted to create an operating system that did not come free, but instead, he needed something that he could customize to fit as per his programming needs. Linux appeared to be his pet project at the start. It was like a side hustle. UNIX is different from the Linux operating system.

Linus built the entire Linux system from scratch. He created Linux because he desired to build an open-source operating system for the people to use. At that time, UNIX was not open-source. People had to pay someone to use UNIX. Similarly, Microsoft was also a paid operating system.

Therefore, Linux came up with the idea of an open-source operating system. He worked up the idea with his friends from the Massachusetts Institute of Technology (MIT). Coupled with building an open-source operating system, they needed an easy-to-use and efficient operating system they could customize to suit their programming needs.

Linux Distributions

When Linus was creating the Linux operating system, he stopped working on it for a while. During that period, he made the code for the operating system public. This allowed everyone to take part in the creation of the system. Scientists and computer geeks started working on the concept as well. They

changed the operating system as they deemed fit.

Major educational institutions and companies liked the concept of this new operating system because everyone who had the source code could install Linux on his or her computer. This is how people started creating different versions of the Linux operating system. Students from the University of California, Berkeley, tried to start creating a version. People from China and people with different occupations also started creating versions to suit their personal needs.

The availability of the source code to the public facilitated the creation of distros or distributions. Distributions are different versions of Linux that people have been creating over time. Linux has different versions, and its many distributions offer it several capabilities. When you have to decide which Linux distribution you need to use, you have to decide what you want your computer to do with Linux. I will explain it by running an analogy with the Windows operating system. When you install Microsoft Windows operating system on our computer, every distribution is built to do things in a particular manner.

There is a version of Linux known as Trustix. Linux Trustix is labeled as the most secure Linux operating system in the market. It is simply a brick. You set up Linux Trustix, and no one will be able to hack it until you do something immensely stupid. There will be no sneaking in by viruses. It is a secure and solid server. However, you have to decide that you really need a secure server before picking up the source code and installing the system.

If you are looking out for a computer that you can use for some office applications, you may need an Ubuntu Linux desktop version. If you are looking for a super-secure computer, then you may need Linux Trustix. If you are looking forward to something that comes with enterprise-level support, you may need to use a Linux distribution paired up with a tech support center to help you out. In this case, Red Hat Linux will be the answer to your needs. Again, you must decide what you want your computer to do to determine the most appropriate Linux distribution to install on your computer system.

Linux is Open-Source

Now that you have learned about the origins of Linux and its distributions, it is time to move forward to the concept of open-source licensing, which makes Linux different from other operating systems. Linux has open-source licensing. You might have heard of open-source software at some point in your life. Open source does not mean that your software is free to use. If you treat all open-source software as free, you will be on the verge of jeopardizing your programming career and company as well. This is legally bad; therefore, we must discuss open-source software to clear the air.

Open-source software means that whenever programmers write a code for a software, they give you the code to see how he or she wrote the program in the first place. It does not mean that the program is free to use. There are different ways by which open-source vendors are paid.

The first way is through the Open Source model, where they give off their software free of cost. However, when you require support or training for the software, they will charge you a certain amount. For example, you can download MySQL server for the Linux server. You find it useful and powerful as well. Even though you have learned the MySQL program's different intricacies, there are some aspects of the software you may need to learn or need support with. Therefore, you approach the software developer, ask for training or support with the software. At this point, you have to pay the programmer or developer for his or her development efforts.

The second way by which developers are paid is through a non-commercial open-source license. This is where most people get into trouble. You have to pay them to use them.

If you want software for home use, there is no problem. Once you use it to connect to a business server, you own a licensing fee to use that software for commercial use. The worst thing is that licensing fees may be over $8,000. It can be that much expensive in some cases. Therefore, it is wise to stay conscious of how you may use the software for non-commercial, personal, or commercial purposes.

The third way by which open-source software programmers are paid is through a paid open-source license. Some of you might ask how software can be on the open-source license if it happens to be a paid software. A paid software is called open-source if the programmer of the software allows you

to see their code.

The fourth way by which these programmers earn money is by recurring license fees for the open-source software. This is like most of the open license programs. They will let you download and test their software free of charge. They would let you see its code as well so that you know how the software works. However, if you want to have the software's legal rights, you will have to pay a yearly or monthly fee for that. This is much cheaper than a one-minute licensing fee that is too much expensive.

The Linux Shell

Now I will move on to the Linux shell of the Linux operating system. The shell of any operating system is the screen by which you interact with that system. Take the example of Microsoft Windows. The Windows shell is its graphical user interface where you can see the mouse pointer at work. You use a pointer to navigate the screen and click on different desktop elements such as icons and folders.

The shell is generally of two types, the first being the graphical user interface (GUI) and the second being the line user interface (LUI). The LUI appears to be as DOS prompts. If you ever have the opportunity to work on the Microsoft DOS prompt, you should know that the screen you see and work on is the line user interface (LUI). It is a black-and-white screen. You see a bunch of commands on the screen to get a specific output from your computer.

Linux is a technical operating system, which is why programmers, engineers, and geeks prefer it. They prefer to use this line user interface because it facilitates them in programming. When you install Linux, you can install the Linux graphical user interface on your system, just like Windows. Here, you can use a mouse to click on things or access a line user interface more suitable for programmers. However, the line user interface on Linux works on a bunch of commands.

You should keep in mind that the line user interface on Linux is more robust than the graphical user interface (GUI). However, when you install Linux with a line user interface for the Linux shell, you see a prompt instead of a mouse. If you do not know what a command prompt is or what you will do

with it, you will most likely be stuck. To help you out, I will give functional examples of Linux commands and shell scripts.

Root

The next concept that you must chew down and digest is that the concept of the root. In the Linux operating system, root relates to the top level of anything. When you work on Linux, you will hear more often about the word ‘root.’ A root user refers to a computer's administrator. A root user is the highest-level user that anyone can be on a computer system. If you can log in as a root user, you can use any command and do anything with the computer. A root user refers to the root of the Linux operating system. It is the point where the operating system is installed on your hard drive. If you think about this in terms of the Windows operating system, C:/ is the operating system root because it is where you have installed the Windows operating system.

A root user has several privileges and the highest level of access that any user can reach. The Linux operating system has many folders, and the home folder is packed up with the user’s data like settings, documents, programs, etc. Therefore, the home directory can be the highest level for a specific user. The most important thing to keep in mind is that when you talk about the root user in the Linux operating system, the root level is the highest level. Root users have complete access to anything in an operating system. Once we move toward typing commands and making the Linux operating system do different tasks, we will realize how important the root user concept is. Where we will be blocked while logged in as a non-root user, we will have access there as a root user.

Reasons to Use the Linux Operating System

The reason you should learn about the Linux operating system is the functionality of the server. The Linux operating system is incredibly rock-solid. Once you have installed the Linux operating system and once you have gone through the quirks and set up the configurations, a Linux operating system will run without overheating and dying in the middle of working. It would run on end. Once you have installed the Linux operating system correctly, it has the power to run for a hundred and fifty days without shutting down.

The Linux operating system is unlike Windows in the sense that you have to reboot it on a weekly basis to avoid certain losses of memory. If you have configured it properly, the Linux operating system would run and do the job with the least concern about the circumstances. There will be little to no operational problems when you install a Linux operating system on your computer.

Installation of the Linux Operating System

In this section, I will walk you through how to install the Linux operating system to try it out and have a feel for the process of using the Linux operating system. I will explain the installation process of the Ubuntu Linux server edition. This is open-source and free to use. Whether you will use Ubuntu for commercial or personal use, there will be no charges. Google and open up the website of Ubuntu and download the ISO file to install the Ubuntu server edition. Burn the file on a disk or load it up on a USB thumb drive. I hope you already know about the burning processor making bootable thumb drive. I am not delving into that to save time.

Since it is a server edition, there will be no graphical user interface on your screen. You will see a line user interface, which in simple words means a black-and-white screen. There will be a blinking cursor on your screen. You have to write commands to move further with your work, or you will be stuck. Let me answer the question that is popping up in your heads as to why the screen is black and white in the Linux operating system's server edition. The biggest reason is safety. The server version of the Linux operating system has a line user interface because every function is like an attack vector for a black hat hacker. Any program that you install on your computer's graphical interface gives a hacker a loophole to enter your system. As there is no graphical interface or external apps in the Linux operating system, the threat of hacking attacks is minimum.

Although the MAC operating system is secure, hackers have deciphered the art of hacking into the Adobe Flash applet that the Mac operating system uses. Over the years, hackers have learned the art of hacking into the Adobe Flash applet that the Mac operating system has been using. Even though the Mac operating system is solid in terms of security, the Adobe Flash has

turned into a security vulnerability. Hackers are now in a position to take over a Mac computer with the help of the Flash software. However, the Linux operating system is free of this vulnerability. Let us move on to the process of installing a Linux operating system on your computer.

The first step for the installation is to download the Ubuntu Linux server operating system. Go to ww.ubuntu.com. On the home page of Ubuntu, you will find different versions of Ubuntu to do several things. You will find a desktop version of Ubuntu. You will also see a netbook version. There will be a cloud version as well, along with a version for tablets and cell phones. You have to locate the server version of Ubuntu and find its link for download.

If you are looking forward to installing Ubuntu on your computer system, you should make sure that the file boots off a DVD and not a hard disk drive. You can do this by pressing the DELETE or F1 key on the keyboard to enter the motherboard's BIOS settings. From there on, you may set up the point where you may want to boot the system from. Several motherboard manufacturers have different ways to enter the BIOS settings, so you should make sure that you consult the user’s manual of the motherboard on how you should do this.

Once you have downloaded the file and made it ready for booting from a CD or DVD, you may proceed to install it. You will be asked in which language you want to install Ubuntu Linux operating system. Choose the desired language and hit Enter. Linux will be installed on your computer system.

Chapter Two

Working with Linux Commands

When we talk about the Linux command line, we refer to the Linux shell. The Linux shell is a program that accepts keyboard commands and passes them on to your operating system for execution. All Linux distributions have a shell program, namely, bash. The bash is the short form of Bourne Again Shell. This is a reference to the fact that bash is the next level version of sh, which was the original UNIX shell program.

When you are using a graphical user interface, you need another program, namely a terminal emulator for interaction with the Linux shell. If you look through the desktop menus, you will be able to locate one.

Let's Start

It is time to start working on the command line of your Linux operating system. The first step that you need to take is to kick off the terminal emulator. Once you have opened it, you should be able to see the following line.

```
linuxbox1:~$
```

This is known as the shell prompt. You will see that whenever the Linux shell is ready to take in the input. It may vary in appearance based on the distribution you are using. If you see a (#) instead of a dollar sign ($), the terminal session will offer you superuser privileges. Either you will be logged in as a root user, or you have selected a terminal emulator that offers superuser privileges. Let us type something on the shell and see what it does for you.

```
linuxbox1:~$ hkhkhhkhk
/bin/sh: hkhkhhkhk: not found
```

If you want to check on your command history, you may check it out by pressing the up-arrow key. Whether you have written and entered a hundred

commands, you will see all of them on the screen. When you press the down-arrow, the command history will disappear. You might be thinking that the command line is all about the keyboard. However, that is not the case. You may use a mouse with the terminal emulator.

The First Commands for Your Linux Operating System

It is time to enter your first commands on your Linux system to see how the terminal works. These are the simplest commands that you will be using to do some common tasks. The first command on the line is the calendar command by which you can display a full calendar of the current month.

```
linuxbox1:~$ cal
      January 2021
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

You can experiment with the date and time command as well. By entering a simple command, you may display the current time and data on your screen. See the following command.

```
linuxbox1:~$ date
Sun Jan 24 10:35:23 UTC 2021
```

When you are using an operating system, you will need to know the exact space on your system to keep the functioning of the system smooth and easy.

```
linuxbox1:~$ df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/root         5120000   2417728   2702272  47% /
devtmpfs           125912         0    125912   0% /dev
tmpfs              126068         8    126060   0% /run
none               126068         0    126068   0% /dev/shm
```

If you want to see the amount of free memory, you will have to enter the free command.

```
linuxbox1:~$ free
               total        used        free      shared  buff/cache   available
Mem:         252140        5252      245980           8         908      243104
Swap:             0           0           0
```

When you have finished your work and you want to end your terminal

session, you simply write exit on the command line and close the terminal session.

Basic Commands

This section of the chapter will walk you through the Unix commands and utilities. The first on the line is the cat command that will display the contents of a file. I will use the system information file and then get back to the shell prompt. This is one of the easiest commands to understand. You can get the output of one or more than one file and read through them.

Note: I will use Fedora distribution to test commands from now onwards.

File Navigation in Linux

Most of you might have been familiar with a graphical user interface. It is very easy to navigate in that system. You see a perfect file tree to see different files on your system. You can open them and navigate through them in seconds. However, that is not the case in Linux. There is no graphical display of files in a Linux operating system. It's better to look at the Linux file system as a maze. The directory in which you are currently present is the current directory. The directory above you is the parent directory. You can display the current working directory with the pwd command.

```
[gha@localhost ~]$ pwd
/root
```

The ls command

The ls command will display the contents of one of the directories in your system. The default is the current directory. You can use ls -1 command to display a lengthier version of the file. You can also use the ls –F command to display the information of the type of files.

```
[gha@localhost ~]$ ls
bench.py  hello.c
```

You can change the current working directory by using the cd command. Just type cd in the shell and the pathname of the directory you want to work in. A pathname is a general route that you take along the branches of a tree to reach the directory you want. You can specify it in two ways. The first way is the absolute pathnames. An absolute pathname starts with the root directory, and

it follows the tree branch by branch until the path to file or desired directory stands completed. Take the example of a directory on your system in which most of the system's programs are installed.

```
[gha@localhost ~]$ cd /usr/downloads
[gha@localhost downloads]$ pwd
/usr/downloads
[gha@localhost downloads]$ ls
```

This command will list the files in that directory. The ls command is one of the widely used commands in a Linux operating system. With this command's help, you can see the contents of a directory and determine the number of files and directory attributes. As you have seen, you can enter the ls command to see the list of files and subdirectories that are contained inside a current working directory.

```
[gha@localhost ~]$ ls /usr
bin  games  include  lib  lib64  libexec  local  sbin  share  src  tmp
```

You can list the home directory of the user and the /usr by using the following command.

```
[gha@localhost ~]$ ls ~ /usr
/root:
bench.py  hello.c

/usr:
bin  games  include  lib  lib64  libexec  local  sbin  share  src  tmp
```

You can add -1 to the ls command to change the format of the display and reveal more details. By adding -1 to the same ls command, you can change the output to a longer format.

Options and Arguments Commands

Most Linux commands are made up of one character preceded by a dash. However, some are longer, with two dashes followed by a word, such as those in the GNU support project. Some commands even allow two or more shorter options to be tagged to one another. In the example you see below, we will give the ls command two options – the first (l) will reduce the long output, while the second (t) sorts the results by modification time.

Let us see the result of the –l command.

```
[gha@localhost ~]$ ls -l
total 8
```

```
-rw-r--r-- 1 root root 114 Dec 26 15:19 bench.py
-rw-r--r-- 1 root root 185 Sep  9  2018 hello.c
```

Now I will add t to –l to prolong the output if it can be prolonged.

```
[gha@localhost ~]$ ls -l
total 8
-rw-r--r-- 1 root root 114 Dec 26 15:19 bench.py
-rw-r--r-- 1 root root 185 Sep  9  2018 hello.c
```

You can reverse the order of the output by using the reverse command. See the syntax of the command as under:

```
[gha@localhost ~]$ ls -lt --reverse
total 8
-rw-r--r-- 1 root root 185 Sep  9  2018 hello.c
-rw-r--r-- 1 root root 114 Dec 26 15:19 bench.py
```

Let’s take a look at some miscellaneous commands that you can pair up with the ls command.

[gha@localhost ~]$ ls is used to produce a list of all the files with proper names that start with a period. It also produces those files that are usually listed as hidden.

[gha@localhost ~]$ ls --all is the long option for the same command.

[gha@localhost ~]$ ls –d is used to specify a directory and display its contents. It does not display the directory. You can pair up this command with the –l command to explore details about the directory instead of the contents.

[gha@localhost ~]$ ls --directory is used to produce more details about the directory or its contents.

[gha@localhost ~]$ ls –F is used to append a particular indicator character to the tail of listed names, like the forward-slash of the name is already in the directory.

[gha@localhost ~]$ ls --classify is used to append the indicatory character to the tail of a listed name.

[gha@localhost ~]$ ls --human-readable is used to display the contents in long format listings and display the files in human-readable formats rather than in the form of bytes.

[gha@localhost ~]$ ls –S is used to sort out the results by the size of the files.
[gha@localhost ~]$ ls –t is used to sort out the results by the modification time.

Finding Out the Type of File

As we begin to explore the system, one of the most useful things you should learn is what is in the files. To do this, we use the file command to help us work out the file type. Linux doesn’t require the filenames to specify the file contents, so, for example, where you would expect a .jpg file to contain an image, Linux doesn’t require this. When the Linux File command is invoked,

it prints out a description of the file contents.

```
[gha@localhost ~]$ file image.jpg
```

There are lots of file types, and, like many other operating systems, Linux treats everything in the system as a file.

The less command

We use the less command for viewing text files. The Linux operating system contains several files with text readable by the human eye, and the less command gives us an easy way to look through them. Text files need to be examined because some of them will be full of system settings. You see these labeled as configuration files, and all of them are stored in a particular format. When you can read the files, you will gain an idea of the way Linux works, and you will find several of the programs used by your system, called scripts, are stored the same way.

Once the less command has been executed, you can scroll through a text file, back or forward. Let's say you are looking at a file defining user accounts for your system. You could use this command:

```
[gha@localhost ~]$ less /etc/passwd
```

Once the command has been given, you can look at the content of a specific file. If it is a long file, use the arrow keys on your keyboard to navigate the contents. Below you can see the other common keyboard shortcuts that help you navigate files with the less command:

- Page Up to scroll back a page
- Page Down to scroll forward a page
- Spacebar to scroll forward a page
- Up Arrow to scroll up a line on the page
- Down Arrow to scroll down the line on the page
- G to go to the end of the text file
- 1G or g to move to the start of the file on the page
- h to display the help screen on the page
- n to search for the next possible occurrence of different

characters

- /characters to search forward to the characters' next occurrence on the page
- q to quit the less command

Your Linux file system is similar to that of other UNIX-like systems, and the Linux File System Hierarchy Standard is the published standard that specifies the design. Below are some of the directories you can explore:

- / is the root directory that plays the role of the starting point of everything.
- /bin is the root directory that contains binaries or programs that ought to be present for a system to boot and run.
- /boot is the root directory containing the Linux kernel. This is the boot loader and contains the drivers needed for the system to boot.
- /var is the root directory which is where data that is most likely to change is stored. You will find a number of pool files, databases, user mail in this directory.
- /var/log is the root directory which contains different log files. It also contains certain records of system activities. The files packed inside the directory are very important. You should monitor it from time to time. The most useful one is /var/log/messages. It is possible that on some systems, you will have to log in as a superuser to access the log files.
- / imp is used to store transient files created by a number of programs. Some configurations may cause the directory to be emptied each time you reboot the system.
- /usr is the largest one in your Linux operating system. It may contain the programs and support certain files that are used by regular users.
- /proc is a special directory because it is not a filesystem used for

storing files on the hard drive. Instead, it is a virtual filesystem the kernel maintains, and the files are snapshots into the kernel, are readable, and will tell you the kernel views a computer.

- /root is the major home directory if you are operating a root account.
- /tmp is used to store transient or temporary files. Be aware that some configurations may empty this directory whenever your system is rebooted.
- /usr/bin carries some executable programs that are installed by the Linux distribution. It is not uncommon for the directory to hold a number of programs.
- /usr/lib is a shared library for different programs that sit in /usr/bin.
- /usr/sbin is loaded with many system administration programs.
- /usr/local is a tree that contains programs not part of the actual distribution but still needed for system-wide use. The programs compiled from the source code go into /usr/local/bin, and you will find this on a new install of the Linux system. However, until the system administrator adds files to it, it will remain empty.
- /usr/share is packed up with shared data that the programs in /usr/bin use. This directory includes different things such as icons, configuration files, sound files, and screen backgrounds.
- /usr/share/doc contains documentation files that are organized by the package.
- /lib is filled up with shared library files that are used by core system programs. These programs are similar to DLLs in the Windows operating system.
- /mnt is used in older Linux operating systems. The /mnt directory carries certain mount points for removable devices in the system. The condition is that the devices should be manually mounted.

- /lost+found is used in case of a partial recovery from the filesystem’s corruption event. This directory stays empty until your system passes through something bad.
- /media is used on modern Linux operating systems. The /media directory contains mount points for the removable media like CD-ROMs, USB drives, and any other device mounted automatically.

Chapter Three

Files & Directories Commands

This chapter will walk you through the commands to navigate through files and directories in the Linux operating system. You can manipulate directories and files. Some of the tasks are easy to do. When you are using commands for navigation, you have more power and flexibility as compared to using a graphical user interface. While the latter option is easy to perform and is suitable to execute simple tasks, you need the command line for pulling off complicated tasks. Copying HTML files from one directory to another is hard in a graphical user interface; however, it is easier to do in a command-line system. Before I move on to exploring the commands that you can use to manipulate files and directories, I will explain some special characters that you can use in Linux.

Special Characters

When you are discussing Linux with others, you may hear about some special characters that Linux users encounter during operations. Each special character has a job to do, knowing which will make things easier for you.

- * is technically called asterisk or star. It is used for a regular expression and as a glob character.
- ‘ is technically called tick or single quote. It is used for literal strings.
- Special character . is technically called dot. It is used to denote the current directory you are working in or the hostname or file delimiter.
- $ is technically called the dollar sign. It is used for denotation of a variable or for representing the end of the line.
- \ is technically called backslash. It is used for macros and literals.

- / is technically called forward slash. It is used as a search command or a directory delimiter.
- ! is technically called bang. It is used for command history and negation.
- | is technically called a pipe. It is used for command pipes.
- _ is technically an underscore or under. It is used for the cheap subtitle for some particular space.
- ` is technically called backtick or backquote. It is used for the substitution of command.
- “ is technically called double quote. It is used for semi-literal strings.
- {} is technically called braces or curly brackets. It is used for ranges or statement blocks.
- ^ is technically called caret. It is used to denote the start of line and negation.
- [] is technically squares or brackets. It is used for ranges.
- ~ is technically called squiggle or tilde. It is used as a directory shortcut and negation.
- # is technically called pound, hash, or sharp. It is used for preprocessor, comments, and substitutions.

Command-Line Editing

As you move around and explore the Linux shell, you will find out that you have the freedom to move in the shell through arrow keys. This comes as a standard on most Linux operating systems. It is a good idea to know about the standard keystrokes on Linux operating systems.

- CTRL-B is used to move the cursor on the screen to the left side.
- CTRL-F is used to move the cursor on the screen to the right side.

- CTRL-E is used to move the cursor on the screen to the end of a line.
- CTRL-A is used to move the cursor on the screen to the start of a line.
- CTRL-P is used to view any previous commands on the screen. It is used to move the cursor upside.
- CTRL-N is used to move the cursor to the downside, and it is also used to view the next command.
- CTRL-K is used to erase everything from the cursor to the end of a line.
- CTRL-W is used to erase the preceding word from the cursor on the screen.
- CTRL-U is used to erase everything from the cursor to the start of a line.
- CTRL-Y is used to paste the erased text that you have already cut off on the screen by the command CTRL-U. It is similar to copy and paste in word processors.

Linux Commands for Directories

The mkdir Linux command is used for the creation of directories. It works as the following:

```
mkdir directoryname
```

The cp command

Another command, the cp command, is used to copy directories and files. You can use it in different ways, such as the following:

```
cp file1 file2
cp files… directoryname
```

You can use the following options with the cp command.

- The cp command option –a or --archive is used to copy the directories and files and their entire attributes, including

permissions and ownerships. Usually, copies take on average default attributes of a user who is performing the copy action.

- The cp command option –u or --update is used when you are copying from one directory to another, and you are copying files that do not exist or newer to the files in the destination directory.
- The cp command option –v or --verbose is used to display the informative messages as you perform the copy.
- The cp command option –i or --interactive is used before overwriting existing files, which prompts the user for confirmation. If you have not specified the option, the cp command will overwrite the existing files, which is not good for your document records.
- The cp command option –r or --recursive is used to copy the contents and files recursively. This particular option is needed when you are copying directories.

Examples of cp command

- The Linux command cp item1 item2 will copy the files of item1 to the files of item2. If item2 exists, it will be overwritten with the contents of item1. If it does not exis, the system will create it.
- The Linux command –i item1 item2 will copy the files of item1 to the files of item2. However, the only exception is that if item2 exists, the user will be prompted before the file is overwritten.
- The Linux command item1 item2 dir1 will copy contents of item1 and item2 into dir1. The directory dir1 must exist before you execute this command.
- The Linux command cp dir1* dir2 uses a wildcard. It will copy all the files of dir1 into dir2. The condition is that dir2 must exist before the execution of the command.
- The Linux command –r dir1 dir2 will copy the directory dir1 to the directory dir2. If dir2 is non-existent, it will be created and will contain the same contents as the directory dir1 will.

The mv command

The mv Linux command performs movement and renaming of files based on how a user uses it. The original filename does not exist after you have executed the command. The mv command is used in the same way as the cp command.

```
mv file1 file2
```

Just like the cp command, the mv command comes with different options that are given as under:

- The mv command option –u or --update is used when you are moving from one directory to another, and you are moving files that do not exist or newer to the files in the destination directory.
- The mv command option –v or --verbose is used to display informative messages as you perform the movement.
- The mv command option –i or --interactive is used before overwriting existing files, which prompt the user for confirmation. If you have not specified the option, the mv command will overwrite the existing files, which is not good for your document records.

Examples for mv command

- the Linux command mv item1 item2 will copy the files of item1 to the files of item2. If item2 exists, it will be overwritten with the contents of item1. If it does not exist, the system will create it. The file item1 will cease to exist in either case.
- the Linux command mv –i item1 item2 will move the files of item1 to the files of item2. However, the only exception is that if item2 exists, the user will be prompted before the file is overwritten.
- the Linux command mv item1 item2 dir1 will move contents of item1 and item2 into dir1. The directory dir1 must exist before you execute this command.
- the Linux command mv dir1* dir2 uses a wildcard. It will move

all the files of dir1 into dir2. The condition is that dir2 must exist before the execution of the command.

- the Linux command mv dir1 dir2 will move the directory dir1 to the directory dir2. If dir2 is non-existent, you will have to create it and move the contents of dir1 into dir2.

The rm command

The rm command is used for removing or deleting directories and files like the following:

```
rm file…
```

In this command file is the name of one or more than one directory and files. Using this command can be highly treacherous. Once you delete something with the rm command, you will never get it back. Linux assumes that you have removed a file on purpose. It does not give it back to you. Therefore, you should use it carefully. Let us say that you want to delete the HTML files in some directory.

So when you are using wildcards with the rm command, you should test the wildcard with the ls command to see the contents of the file you are going to remove from the system. You can use the arrow key to recall the command and replace it with rm then.

- The mv command option –r or --recursive is used to delete directories recursively. This means that if the directory you are deleting has subdirectories, you can delete them as well. If you want to delete a directory, you should specify this option.
- The mv command option –f or --force is used to ignore the non-existent files. It does not prompt. It tends to override the –interactive option.
- The mv command option –v or --verbose is used to display the informative messages as you perform the deletion.
- The mv command option –i or --interactive is used before deleting the existing files, which prompts the user for confirmation. If you have not specified the option, the rm

command will silently delete the existing files, which is not good for your document records.

Examples of rm command

- The Linux command rm item1 will silently delete files of item1.
- The Linux command rm –i item1 will delete the files of item1. However, the only exception is that the user will be prompted before the file is deleted.
- The Linux command rm item1 dir1 will delete the contents of item1 and dir1. The directory dir1 must exist before you execute this command.
- The Linux command rm –rf item1 dir1 works the same as above, with the exception that if both item1 and dir1 do not exist, the rm command will go on silently.

Chapter Four

Practical Work with Commands

You have gone through a series of commands. Each has some mysterious options and arguments. This chapter will walk you through some more commands to make you more familiar with the Linux command line.

The type command

This is a built-in Linux command that displays a typical type of command that the shell is going to execute. See how you can use it.

```
[gha@localhost ~]# type type
```

type is a shell builtin

```
[gha@localhost ~]# type ls
```

ls is aliased to `ls --color=auto'

```
[gha@localhost ~]# type cp
cp is /bin/cp
[gha@localhost ~]# type mv
mv is /bin/mv
[gha@localhost ~]# type rm
rm is /bin/rm
[gha@localhost ~]# type cd
```

cd is a shell builtin

```
[gha@localhost ~]# type help
```

help is a shell builtin

You can see that there are different results of all these different commands.

The which command

Sometimes more than one version of executable programs is installed on a particular system. While this is not common on desktop systems, it is not quite unusual on large servers. You can use the 'which' command to know the exact location of an executable.

```
[gha@localhost ~]# which ls
ls='ls --color=auto'
        /bin/ls
```

The which command works well for executable programs only and not for built-ins or alias that are mostly substitutes for executable programs. When you try to use the which command on shell built-in, you get either no response or a big error message.

```
[gha@localhost ~]# which help
/usr/bin/which: no help in (/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/l
local/bin)
```

Documentation of Commands

You can take a step further and fish out the available documentation in the system for each type of command.

The help command

Bash shell has a built-in help facility for shell built-ins. If you want to use it, you can type help and the name of the shell built-in. See the following example.

```
[gha@localhost ~]$ help cd
cd: cd [-L|[-P [-e]] [-@]] [dir]
```

Change the shell working directory.

Change current directory to DIR. The default DIR is the HOME Shell variable's value.

The CDPATH variable is used to define the search path leading to the directory where DIR is. CDPATH indicates alternative file names by using a colon (:) to separate them. A null name indicates the current directory, and where a slash (/) precedes a directory name, then CDPATH isn't used.

The details are longer than I have mentioned here. I cut it down to save space. You can type the command on the terminal and see how the help command can help you out when you get stuck as you operate the Linux operating system. When you see square brackets in the description of a command's syntax, you should keep in mind that they indicate only optional items. If you see a vertical bar character in the description, it denotes mutually exclusive items. Some executable programs support the --help option. When you apply

the option, it displays a lengthy description of the command's supported syntax and the related options.

```
[gha@localhost ~]$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
```

Create the DIRECTORY(ies) if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

```
-m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
```

The man command

Most executable programs intended for command-line use offer a formal piece of documentation known as a man page or a manual page. A special paging program known as man is loaded up on Linux to view the details. You can type the man command and then the title of the command. The man pages differ in format and contain a title or a synopsis of the command's syntax, a listing, and a description of the purpose of the command. The man page does not include examples, and they are intended as a special reference and not as a tutorial. See the following example.

```
[gha@localhost ~]$ man ls
LS(1)       User Commands          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION

       List information about the FILEs (the current directory by default).

       Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

On most of the Linux operating systems, man uses less for displaying the manual page so that all familiar less commands work while the page is displayed. The manual that man displays has different sections, and it covers user commands and system administration commands. In addition to that, it also covers file formats, programming interfaces, and much more. See the complete layout of the manual as under:

- The number 1 section of the man command will display user commands.
- The number 2 section of the man command will display programming interfaces for the kernel system calls.
- The number 3 section of the man command will display programming interfaces to the C library.
- The number 4 section of the man command will display special files such as the device nodes and the drivers.
- The number 5 section of the man command will display formats of the files.
- The number 6 section of the man command will display the details of games and entertainment like screensavers.
- The number 7 section of the man command will display miscellaneous items.
- The number 8 section of the man command will display the commands related to system administration.

At times, you may need to look into a specific section of a manual to find out what you are looking for. If you do not specify a section number, you will always get the very first instance of a specific match in section 1.

The apropos command

It is possible to search the list of man pages for some possible matches based on a specific search term. This crude approach is often helpful. The syntax is as under:

```
[gha@localhost ~]$ apropos floppy
```

The whatis command

The whatis command in Linux operating displays the name and a short description of a man page. The syntax is as under:

```
[gha@localhost ~]$ whatis ls
```

The man pages that come with a Linux distribution system are labeled as a

reference documentation instead of a tutorial. A number of man pages are tough to read, so tough that a majority of us skip them right away without giving them a second glance. However, the most difficult of the man pages are of bash. They are long, complicated, and almost unreadable. However, their length becomes their strength as they contain plenty of useful information for you to add to your knowledge. When you have gone through one page, it will start making sense to you.

The info command

The GNU project provides info pages as an alternative option to the man pages. A reader program called info displays these pages, and these are hyperlinked in the same way web pages are. For example:

```
[gha@localhost ~]$ info ls
```

Next: dir invocation, Up: Directory listing

10.1 ‘ls’: List directory contents

==========

In the ‘ls’ program, you will find information about files of any type, and that includes directories. As usual, arbitrary mixing of options ad file arguments is accepted.

By default, ‘ls’ will list the contents of non-option command-line arguments that are also directories, but this is not a recursive list, and files starting with ‘.’ are left out.

The info program will read the info files – these are structures that go into individual nodes, each containing one topic. Hyperlinks in the info files help users move between the nodes, and these hyperlinks are identified by a leading asterisk. Typically, activating these hyperlinks requires the user to click them with the cursor and press Enter. The info command provides this information:

- The ? key in the info menu will display the command help.
- The Page up or Backspace keys in the info menu will display the previous page.
- The Page down or Spacebar keys in the info menu will display

the succeeding page.

- The n key in the info menu will display the next node in the menu.
- The p key in the info menu will display the previous node in the menu.
- The u key in the info menu will display the parent node of the node that is presently on display.
- The Enter key in the info menu will allow you to follow the hyperlink at the cursor's present location.
- The q key in the info menu will allow you to quit the menu.

Chapter Five

Redirection Commands & Keyboard Tricks with Linux Commands

In this chapter, we will talk about the coolest feature of the command line, which is called I/O redirection. The I/O stands for input/output. With this facility, you may redirect the input and output of commands and connect multiple commands to make a powerful command known as pipelines.

The output of a program is of two types. First, you have the result of the program which contains the data program produces. Secondly, there are status and error messages that instruct you how the program is getting along. If you look at the command ls, you will see that it delivers results and error messages on your screen.

This program sends the output to another file called standard error. Both standard error and output are connected to the screen, and they are saved into a disk file. Many programs take input from a facility known as standard input stdin, which is attached to a keyboard.

I/O redirection allows you to change where the output goes and where the input flows in from. Usually, output goes to the screen, and the input flows in from the keyboard in Linux; I/O has the power to change that.

Redirection

I/O redirection helps you in redefining the direction of the output. To redirect the standard output to a file instead of the screen, you have to use the redirection operator that is followed by the name of the file. It is useful more often to store the output of a certain command inside a file. You may tell the shell to direct the ls command's output to a file namedls-output.txt instead of the screen.

```
[gha@localhost ~]$  ls -1 /usr/bin > ls-pt.txt
```

I have created a long listing of /usr/bin directory and then sent the results to the file ls-pt.txt.

Keyboard Tricks

Unix is an operating system for people who love to type. The command line does not allow a mouse to operate. Linux commands can be exhaustive, therefore, you should learn a bunch of keyboard tricks.

Text Modification

The following list shows how you can modify text in the Linux command line. The terms killing and yanking refer to cutting and pasting, respectively.

- The keyboard trick CTRL-D will help you delete the character that is at the present position of the cursor.
- The keyboard trick CTRL-T will help you transpose the character at the present location of the location with the one that is preceding it.
- The keyboard trick Alt-U will help you convert into uppercase the characters from the cursor's present position to the ending point of the word.
- The keyboard trick Alt-L will help you convert into lowercase the characters from the cursor's present position to the ending point of the word.
- The keyboard trick Alt-T will help you transpose the words at the cursor's present position with the preceding one.
- The keyboard trick CTRL-K will help you kill text from the cursor's present position to the ending point of a line.
- The keyboard trick CTRL-U will help you kill text from the cursor's present position to the starting point of a line.
- The keyboard trick ALT-D will help you kill text from the cursor's present position to the end of a current word.
- The keyboard trick ALT-Backspace will help you kill text from

the cursor's present position to the starting point of the current word. If the cursor is presently at the start of a word, it will also kill the previous word.

- The keyboard trick CTRL-Y will help you yank text from the kill-ring and paste it at the cursor's present position.

Completion Commands

Another way by which the shell will help you is through completion. This occurs when you hit the Tab key while you are typing the command. When you are halfway through a command, you can enter tab to complete the command. The important thing to remember is that you should not hit the Enter key.

Completion commands

- The keyboard trick Alt-$ will help you display the list of all possible completions. You may also do this by pressing the Tab key twice. This is much easier to do.
- The keyboard trick Alt-* will help you insert the possible completions. This is highly useful when you intend to use more than one match.

Searching History

Bash maintains a history of the commands you type in it. The list of commands is kept in the home directory in a file named .bash_history. The history facility is highly useful for cutting down on the amount of typing that you need to do especially when you combine it with command-line editing. Here is the syntax of the history command.

```
[gha@localhost ~]$ history | less
```

Bash, by default, stores the last 500 commands that you have entered. Suppose you want to find the commands that you used to list /usr/bin. Here is the way to do that.

```
[gha@localhost ~]$ history | grep /usr/bin
```

There is a list of keystrokes that you may use when you are navigating

through the history command.

- You can use the keyboard shortcut CTRL-P to move to the latest history entry. This performs the same action as the up arrow.
- You can use the keyboard shortcut CTRL-N to move to the next history entry. This performs the same action as the down arrow.
- You can use the keyboard shortcut ALT-> to move to the bottom of the list of history that is the current command line.
- You can use the keyboard shortcut ALT-< to move to the starting point of the list of history.
- You can use the keyboard shortcut CTRL-R to reverse the incremental search. This option incrementally searches from the current command line up the list of history.
- You can use the keyboard shortcut CTRL-O to execute the present item in the list of history and then move on to the next one. This is handy if you are looking forward to re-execute a sequence of commands in the list.
- You can use the keyboard shortcut ALT-N to forward the search. This is non-incremental.
- You can use the keyboard shortcut ALT-P to reverse the search. This also is non-incremental. Coupled with this keyboard shortcut, you can type the search string and then press ENTER before you have performed the search.

History Expansion

The shell offers a special type of expansion for different items in the history list by using the ! character. We already have seen how a number may follow the exclamation point to insert a particular entry from the list of history. There are many other expansion features. History expansion mechanism has many available elements, but this subject is too arcane.

- You can type the sequence !! to repeat the latest command.

However, it may be easier to press the up arrow and then hit ENTER.

- You can type the sequence !?string to repeat the latest history list item that contains a string.
- You can type the sequence !string to repeat the latest history list item that starts with a string.
- You can type the sequence !number to repeat the latest history list item.

Chapter Six

Process Commands

Today's operating systems are capable of multitasking, which means they can do more than one thing at a time. They switch rapidly between executing programs, and the Linux kernel manages this. Linux uses processes to organize the programs queued for execution in the CPU.

Sometimes, a computer system can become slow, or a specific application might stop working. This chapter is designed to show you the command-line tools you can use to terminate certain processes that don't work properly during the system operations.

When your system wakes, certain activities are activated as processes by the kernel. This is followed by a program called init being launched. This program runs shell scripts known as init scripts, which are used to start the system services. Lots of these services are implemented as daemon programs, working in the background and doing what they have to do without a user interface. You can find the processes command here:

```
[gha@localhost ~]$ ps
PID TTY          TIME CMD
   48 hvc0     00:00:00 sh
   82 hvc0     00:00:00 ps
```

The result lists process that are sh and ps respectively. If we add an option to the command ps, we can detailed result on our screens. See the following example.

```
[gha@localhost ~]$ ps x
  PID  TTY      STAT   TIME COMMAND
    1 ?        S      0:00 /bin/sh /sbin/init
    2 ?        S      0:00 [kthreadd]
    3 ?        I      0:00 [kworker/0:0]
    4 ?        I<     0:00 [kworker/0:0H]
    5 ?        I      0:00 [kworker/u2:0]
    6 ?        I<     0:00 [mm_percpu_wq]
    7 ?        S      0:00 [ksoftirqd/0]
    8 ?        S      0:00 [kdevtmpfs]
    9 ?        I<     0:00 [netns]
```

```
 10 ?      S      0:00 [oom_reaper]
 11 ?      I<     0:00 [writeback]
 12 ?      I<     0:00 [crypto]
 13 ?      I<     0:00 [kblockd]
 14 ?      S      0:00 [kswapd0]
 15 ?      I      0:00 [kworker/0:1]
 33 ?      S      0:00 [khvcd]
 43 ?      Ss     0:00 dhcpcd
 48 hvc 0     Ss     0:00 sh -l
 71 ?      I      0:00 [kworker/u2:1]
102 hvc0     R+     0:00 ps x
```

The x option informs ps that it needs to show all the processes, no matter what terminal controls them. Because the system is running many processes, a long list is produced, and more often than not, it is helpful to pipe the ps output into less and view them that way. Some combinations of options produce many output lines, and it is a good idea to maximize the emulator window in the terminal.

Process States

Here is a rundown of process states.

- The process state R denotes running. This process is either running or is ready to run in the system.
- The process state S denotes Sleeping. The process is in Sleep mode and not running. It is waiting for a certain event to start, such as a keystroke or network packet.
- The process state Z denotes the zombie process. This is a child process that has been terminated, but the parent process has not cleaned it up.
- The process state D denotes Uninterruptable sleep. This process is waiting for the I/O like a disk drive.
- The process state T denotes stopping. This process is instructed to stop.
- The process state N denotes a low-priority process. It’s a good process that will obtain the processor time when all other high-

priority processes have been serviced.

- The process state < denotes a high priority process. It is possible to give away more importance to a process and allowing it more time on your CPU. A process that has higher priority is less nice because it takes more of the CPU's time.

Other characters may follow this process and these indicate characteristics of other processes. The aux command can be used with ps to provide another set of options:

```
[gha@localhost ~]$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  1.3   3136  2516 ?        S    14:35   0:00 /bin/sh /sbin/i
root         2  0.0  0.0      0     0 ?        S    14:35   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I    14:35   0:00 [kworker/0:0]
root         4  0.0  0.0      0     0 ?        I<   14:35   0:00 [kworker/0:0H]
root         5  0.0  0.0      0     0 ?        I    14:35   0:00 [kworker/u2:0]
root         6  0.0  0.0      0     0 ?        I<   14:35   0:00 [mm_percpu_wq]
root         7  0.1  0.0      0     0 ?        S    14:35   0:02 [ksoftirqd/0]
root         8  0.0  0.0      0     0 ?        S    14:35   0:00 [kdevtmpfs]
root         9  0.0  0.0      0     0 ?        I<   14:35   0:00 [netns]
root        10  0.0  0.0      0     0 ?        S    14:35   0:00 [oom_reaper]
root        11  0.0  0.0      0     0 ?        I<   14:35   0:00 [writeback]
root        12  0.0  0.0      0     0 ?        I<   14:35   0:00 [crypto]
root        13  0.0  0.0      0     0 ?        I<   14:35   0:00 [kblockd]
root        14  0.0  0.0      0     0 ?        S    14:35   0:00 [kswapd0]
root        15  0.0  0.0      0     0 ?        I    14:35   0:00 [kworker/0:1]
root        33  0.0  0.0      0     0 ?        S    14:35   0:00 [khvcd]
root        43  0.0  0.7   1944  1472 ?        Ss   14:35   0:00 dhcpcd
root        48  0.0  1.5   6204  2940 hvc0     Ss   14:35   0:00 sh -l
root        71  0.0  0.0      0     0 ?        I    14:35   0:00 [kworker/u2:1]
root       160  0.0  1.4   8180  2740 hvc0     R+   15:02   0:00 ps aux
```

This command displays the processes that belong to all users of a system. When you use the options without a leading dash, it invokes the command with BSD-style behavior. Here are the details of BSD-Style ps Column Headers.

- The BDS-style ps column header, namely USER alludes to User ID. This denotes the owner of a process.
- The BDS-style ps column header called RSS indicates Resident Set Size. This denotes the amount of physical memory (RAM) used by the process.
- The BDS-style ps column header, namely VSZ alludes to the

virtual memory size of the system.

- The BDS-style ps column header, namely %CPU alludes to the CPU usage in percentage.

Using the Top Command to View Processes

The ps command reveals a lot about what your Linux machine is doing. However, it provides you just a snapshot of the state of the machine as the ps command is executed in the system. To see a dynamic view of the machine's activity, I will use the top command.

```
[gha@localhost ~]$ top
top - 15:10:01 up 34 min,  0 users,  load average: 0.05, 0.05, 0.02
Tasks:  20 total,   1 running,  19 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.9 us,  4.9 sy,  0.0 ni, 93.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :    182.0 total,    171.2 free,      4.9 used,      5.8 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.    172.1 avail Mem


  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  180 root      20   0    8424   3084   2584 R   7.4   1.7   0:01.82 top
    1 root      20   0    3136   2516   2148 S   0.0   1.4   0:00.95 init
    2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/0+
    4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0+
    5 root      20   0       0      0      0 I   0.0   0.0   0:00.00 kworker/u+
    6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu+
    7 root      20   0       0      0      0 S   0.0   0.0   0:02.45 ksoftirqd+
    8 root      20   0       0      0      0 S   0.0   0.0   0:00.10 kdevtmpfs
    9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
   10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reaper
   11 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 writeback
   12 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 crypto
   13 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kblockd
   14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kswapd0
   15 root      20   0       0      0      0 I   0.0   0.0   0:00.03 kworker/0+
   33 root      20   0       0      0      0 S   0.0   0.0   0:00.00 khvcd
   43 root      20   0    1944   1472   1196 S   0.0   0.8   0:00.22 dhcpcd
   48 root      20   0    6204   2940   2380 S   0.0   1.6   0:01.04 sh
   71 root      20   0       0      0      0 I   0.0   0.0   0:00.01 kworker/u+
```

As you enter the top command in the command line, you will see that the figures are continuously updating. They update every three seconds.

Chapter Seven

Working with Linux Editors

Before moving on to Linux shell scripting, we should make ourselves acquainted with the text editors that we will have to use to write shell scripts. This chapter will walk you through the features like searching, copying, cutting, and pasting. The more practice you do to learn an editor, the faster you will learn to write shell scripts in Linux.

The Vim Editor

If you are working in the command line mode, you may need to become familiar with a text editor that will be operating in a Linux console. The vim editor is the original editor that Unix uses. It makes use of the console graphics mode for the emulation of a text-editing window, which allows you to see different lines of the file, move around across the files, and edit, insert or replace a piece of text. The vim editor works well with the data that is in a memory buffer. You have to type vim and the name of the file that you have to edit to open the editor with the desired file.

If the editor is started without a filename being supplied, it opens but with no file. The vim editor detects the session's terminal type and uses full-screen mode so the console window can use the editor area. The initial window will show the file contents and a message line at the bottom of the window. If the contents don't take up the entire screen, a tilde is placed on the lines excluded from the file. The vim editor has two operational modes – normal and insert mode. When you open a file for editing, vim goes into normal mode, and certain keystrokes are interpreted as commands.

In the insert mode, each key typed at the cursor is inserted into a buffer. To get into insert mode, press the i key, and to get out of it and back to normal mode, press the ESC key.

You can use the arrow keys in normal mode to move around the text so long

as vim has detected the correct terminal type. The vim command includes those for cursor navigation, as you can see below:

- The command h in vim Linux editor is used to move to the left by one character.
- The command l in vim Linux editor is used to move to the right by one character.
- The command j in vim Linux editor is used to move down the cursor by one line.
- The command k in vim Linux editor is used to move up the cursor by one line.
- The command PageDown or Ctrl-f in vim Linux editor is used to move forward one screen of your data.
- The command PageUp or Ctrl-b in vim Linux editor is used to move forward one screen of your data.
- The command gg in the vim Linux editor is used to shift to the first line in the buffer.
- The command G in vim Linux editor is used to move to the last line in the buffer.
- The command num G in vim Linux editor is used to move to the line number in the buffer.

The vim editor carries a special feature in the normal mode that is known as the command line mode. The command line mode offers an interactive command line where you may enter additional commands to control different types of actions in the vim editor. To get to the command line mode in the vim editor, you have to press the colon key while you are in the normal mode. The cursor will move to the message line. Here you will see a colon popping up on the screen which indicates that you should now enter a command.

When you are in the command line mode, you may enter several commands to save the buffer to file and move out of the editor.

- The command q in vim Linux editor is used to exit the system if you have made no changes to buffer data.
- The command wq in vim Linux editor is used to save buffer data to file and then exit the editor.
- The command q! in vim Linux editor is used to quit the editor and then discard the changes that are made to the data of buffer.
- The command w *filename* in vim Linux editor is used to save your file with a different filename.

Data Editing

When you are in the insert mode, you may insert data in the buffer. Sometimes you have to add or remove some data after entering into the buffer. While you are in normal mode, the vim editor will provide certain commands for editing data in the buffer. You should take extra care when you try to use a PC keyboard Delete or Backspace keys while you are inside the vim editor. The vim editor recognizes the keyboard Delete key as an x command functionality, which will delete a character at the current cursor location.

Copy & Paste

A standard feature of these editors is their ability to make a cut or copy data, and after that, paste it into the document. Cutting and pasting text is easy. When vim removes your data, it will forward it into a separate register. You can then retrieve that data by using the p command. You can use the dd command to remove different lines of text. When you try to copy text, you may find it a bit trickier.

- x in vim Linux editor is used to delete a character at the cursor's current position.
- R *text* in vim Linux editor is used to overwrite the data at the cursor's current position with the text until you hit the Escape key.
- r *char* in vim Linux editor is used to replace any single character

at the cursor's current position with char.

- dd in vim Linux editor is used to delete a line at the cursor's current position.
- A in vim Linux editor is used to append data to the end point of the line at the cursor's current position.
- a in vim Linux editor is used to append data next to the cursor's current position.
- J in vim Linux editor is used to delete a line break to the end of the line at the cursor's current position.
- dw in vim Linux editor is used to delete a word at the cursor's current position.
- d$ in vim Linux editor is used to delete the text to the ending point of a line from the cursor's current position.

The KDE Editor Family

If you are using a Linux distribution that uses KDE desktop, you will see a couple of options in relation to text editors. The KDE project will officially support a couple of editors, such as Kate and KWrite. Both are graphical text editors containing a number of advanced features and some extra niceties that are not found in some common standard editors. This section will describe each editor and also the features that you may use in shell scripting.

The KWrite Editor

This is the basic editor for the KDE environment. It offers word-style text editing with support for code syntax editing and highlighting and is used for different types of programming languages. It uses color-coding to distinguish comments, constants, and functions. You should notice that the for loop carries an icon that connects the opening and closing braces. The editing window provides cut-and-paste capabilities.

The editor has many command line parameters you may use to customize how it starts.

- The Kwrite editor command --stdin triggers the editor to read the data from a standard input device in place of a file.
- --column triggers the editor to specify a column number in a file to initiate the process in the window of the editor.
- --encoding triggers the editor to specify the type of character encoding for a file.
- --line triggers the editor to specify the number of a line in a file to initiate in the window of the editor.

The KWrite editor offers a toolbar and a menu bar at the top of the window used for editing, which allows you to select different features and then change the editor's configuration settings. The menu bar will contain the following items.

- File helps you load up, save, export or print different pieces of text from different files.
- Edit aids you in manipulating text in buffer.
- View helps you manage the appearance of the text in the editor of the window.
- Bookmarks helps you handle the pointers to get back to certain locations in the text.
- Tools has some specialized features for manipulation of the text.
- Settings helps you configure the way your editor handles the text.
- Help will give you the information about the editor and its commands.

The Edit Menu of KWrite Editor

Here is a rundown of different commands of the Edit menu of the KWrite editor.

- Undo is used to reverse your latest action in the editor.

- Redo is used to reverse the latest action you undid.
- Cut is used to delete the text that you have selected and then placed in the clipboard.
- Copy is used to copy any piece of text that you have selected to the clipboard.
- Copy as HTML is used to copy the selected text to the clipboard as the HTML code.
- Paste is used to insert the current content of the clipboard at the present position of the cursor on the screen.
- Select All is used to select the entire text in your editor.
- Deselect is used to deselect the piece of text that you have currently selected.
- Block Selection Mode permits you to select a piece of text between the columns instead of all the lines.
- Overwrite Mode is used to toggle between insert mode and overwrite mode by replacing text with newly typed text instead of inserting the new text.
- Find is used to produce the Find Text dialog box, which allows you to customize the text search.
- Find Next is used to repeat the last find operation forward in the buffer.
- Find Previous is used to repeat the last find operation backward in the buffer.
- Replace is used to bring about on the screen the Replace With Dialog box. You can use it to customize a text search.
- Go to Line is used to produce the Goto dialog box. It allows you to enter a certain line number. The cursor jumps to a specified line.

The KWrite Editor Tools

KWrite offers a wide range of tools for users to write and edit their shell scripts. Some of the tools are as under:

- The KWrite editing tool, Read Only Mode will lock the text so that no changes may be made while you are still using the editor.
- Filetype will select the file-type scheme used in a piece of text.
- Spelling will start the spellcheck program at the starting point of a piece of text.
- Highlighting will highlight the text based on the content like program code or the configuration file.
- Indent will increase the indentation in a paragraph by one.
- Indentation, will automate indentation based on the selection you have made in the editor.
- Spelling will initiate the spellcheck program at the beginning point of the text.
- Spellcheck Selection will initiate the spellcheck program on the selected section of a piece of text.
- Unindent will cut down on the indentation of a paragraph by one.
- Clean Indentation will return the paragraph indentation to the original settings.
- Align will bring back the selected lines' current line to the default indentation settings.
- Uncomment eliminates a comment symbol from the present line based on the type of file.
- Lowercase, will set the selected text or the character at the cursor's present position to lower case.
- Uppercase, will set the selected text or the character at the cursor's present position to upper case.

- Capitalize, will set the first letter of the selected text or the word at the cursor's present position to uppercase.
- Join Lines will pair up the selected text or lines at the present position with the next line into a single line.
- Word Wrap Document will enable word wrapping inside a piece of text. If a line moves past the editor window edge, it will continue on the next line.

The GNOME Editor

If you are operating a Linux operating system through the GNOME editor, you will see a graphical text editor that you can use easily and well. It is a basic text editor that has a couple of advanced features for the fun of editing. When you start gedit with multiple files, it will load the files into individual buffers and display each of them as a tabbed window inside the editor's main window. The left frame inside the gedit editor will show the documents that you have been editing.

Basic Features of the Editor

Besides a window for editing, gedit uses a menu bar and a toolbar that allows you to set up the configure and feature settings. The toolbar provides some real quick access to the menu bar items, which are as under:

- The menu item File helps you load up new files, save existing files, and print different pieces of text from different files.
- Edit aids you in manipulating text in the buffer and for setting up the editor preferences.
- View helps you set up features of the editor related to the display. It also works on setting up the text-highlighting mode.
- Search helps you find and replace pieces of text in the editor's active buffer spot.
- Tools is used to access the plugin tools that are installed in gedit.
- Documents helps you manage different files that are open in buffer areas.

- Help will give you information about the editor and its commands. You will see a complete manual to use the editor and work with the commands.

Preferences

The Edit menu offers you the Preferences item, which allows you to customize the operations of the editor. The Preferences dialog box carries tabbed areas that allow you to set up the features of the editor as per your requirements.

View

The View tab offers many options for displaying the text in the editor window.

- The first option you will find in the View tab is named Text Wrapping. It determines how you can handle long pieces of text in your editor. When you enable the option, it wraps up long pieces of text to your editor's next line. The Do Not Split Words Over Two Lines option bars the auto-inserting of hyphens into long words to prevent the same from splitting in between the two lines.

- The second option you will find in the View tab is named Line Numbers. This option displays the numbers of lines in the left margin in the window of the editor.

- The third option you will find in the View tab is Current Line, highlighting a line where the cursor is positioned. This enables you to find out the cursor position easily.

- The fourth option you will find in the View tab is named Right Margin, which enables the right margin. This option also allows you to set up the number of columns in the editor window. The default value of the columns is 80.

- The fifth option you will find in the View tab is named bracket matching. When you enable this option, it will allow you to easily match brackets when you are writing if-then statements, for loops or while loops, and any other code lines that involve

brackets.

The line-numbering and matching brackets offer a handy environment to users. This is easy for troubleshooting and is not found in many other text editors.

Editor Tab

The Editor tab offers options to handle indentation and tabs. It also oversees how you save the document after editing.

- The Tab Stops option in the Editor tab allows you to set up the number of spaces that should be skipped while you hit the Tab key. The default value of skipping is 8. You can reset that. The feature also has a checkbox that inserts spaces rather than tab spaces.
- The second option you will find in the Editor tab is named Automatic Indentation. When you enable that, it allows the gedit to automate line indentation in the text for code elements and paragraphs like if-then statements and the loops.
- The third option you will find in the Editor tab is named File Saving, which offers two features to save files. You have the option to create a backup copy of the file, which has been opened in the edit window. You can also set up the option whether you like to save the file at a preselected interval or not.

Font & Colors

The Font & Colors option offers a configuration of two items.

- The Font option in the Font & Colors tab allows you to select Monospace 10 as a default font or select a customized font style and size from a dialog box.
- The Colors option in the Font & Colors tab allows you to select a default color scheme that is used for text, selected text, background, and selection colors. It is used to choose a custom color for different categories in the list.

The default colors match the standard desktop theme of GNOME, which has been selected for the desktop. These colors change to match the desktop theme when you select a different color.

Syntax Highlighting

The Syntax Highlighting tab gives options for the configuration of how gedit highlights different elements in programming mode. The gedit editor has the power to detect what programming language exists in a text file. It automatically set up the appropriate syntax highlighting for that text. Also, it allows you to customize the highlighting feature by selecting your favorite colors to highlight different elements of the code. The elements change based on the programming code type that you have selected. For the shell scripts, you may select the sh highlighting mode, which contains certain color schemes.

Plugins

The Plugins tab offers control over the plugins that are used in gedit. Plugins are individual programs that tend to interface with gedit and provide added functionality. Many plugins are available for gedit. However, not all of them are installed in the system by default. The plugins that are enabled show a checkmark in a checkbox that is next to their names. Some plugins like External Tools plugin provide some additional configuration features after you have selected them. It allows you to set up a shortcut key to start the terminal where gedit displays the output.

- You can use the plugin Change Case to change the case of a piece of selected text.
- You can use the plugin Document Statistics to report the number of words, characters, lines, and non-space characters.
- You can use the plugin Insert Date/Time to insert the current time and date in multiple formats at the cursor's present position.
- You can use the plugin External Tools to provide a particular shell environment in the editor to execute scripts and commands.
- You can use the plugin Indent Lines to provide un-indentation and advanced line indentation.

- You can use the plugin File Browser Pane to provide a file browser to ease off the process of file selection and editing.
- You can use the plugin Tag List to provide a particular list of commonly used strings that you can easily enter into the text.
- You can use the plugin Modelines to provide emacs-style messages to the bottom side of your editor's window.
- You can use the plugin Python Console to provide interactive consoles to the bottom side of the editor window to enter commands with the Python programming language's help.
- You can use the plugin Spell Checker to provide proper dictionary spellchecking for a text file.
- You can use the plugin Snippets to store some often-used pieces of text for retrieval in the text.
- You can use the plugin Sort to sort out an entire file or a piece of selected text.

The emacs Editor

Emacs is a popular editor that existed before Unix did, and developers loved it because it could be ported into Linux. It began life as a console editor, similar to vi, but soon migrated to the graphical environment. It provides access to the original console editor and uses a graphical X windows window for text editing in graphical environments.

When the emacs editor is started from the command line, it determines whether an active X Window session is present and starts in graphical mode. The console mode version uses multiple commands that you need to learn to edit a program. Certain key combinations are required using the CTRL and Meta keys on the keyboard. Typically, the Meta key is mapped to a PC’s ALT key, although this may not be the case on all systems. Abbreviations are used – CTRL is abbreviated as C- and Meta as M-.

Basic Commands of emacs Editor

To edit an emacs file from the command line, the following command is

required:

```
$ emacs myfile.c
```

When the emacs console window is opened, you see a short introduction and a help screen. When a key is hit, a file is loaded in the buffer, and the text is displayed. You see a standard menu bar at the top of the window, but this cannot be used while in console mode, only in graphical mode. Unlike the vim editor, the emacs editor has a single mode, where you move into and out of insert mode to flick between inserting commands and text. When you type printable characters, emacs will insert them at the cursor's current position, and when a command is typed, it is executed. To move the cursor around the buffer, the arrow keys and the PageUp and PageDown keys are required. Other commands needed to move the cursor include:

- You can use the C-p command to move up the previous line in the text.
- You can use the C-n command to move down the next line in the text.
- You can use the C-b command to move back or to the left side by one character.
- You can use the C-f command to move forward or to the right side by one character.

If you are looking forward to making longer jumps, you can use the following commands.

- You can use the M-f command to move to the right side onto the next word.
- You can use the M-b command to move to the left side, back to the previous word.
- You can use the M-a command to move to the start of the present sentence.
- You can use the M-e command to move to the ending point of the present sentence.

- You can use the C-e command to move to the ending point of the present sentence.
- You can use the C-a command to move to the starting point of the present sentence.
- You can use the M-> command to move to the ending line of the text.
- You can use the M-< command to move to the starting line of the text.
- You can use the M-v command to move back by a single screen of the data.
- You can use the C-v command to move forward by a single data screen.

You can save the editor buffer as a file and exit the emacs editor using the following commands:

- You can use the C = x C = s command to save what you have written in the current buffer into a file.
- You can use the C = x C = c command to exit the emacs editor and stop the operations of the editor.
- You can use the C = z command to exit the emacs editor. However, the command does not keep the session running, so that you may come back to that.

Editing Data

The emacs editor is powerful in terms of inserting and deleting text. Inserting text simply requires that you move your cursor to where you want the text inserted and type the text. The backspace key is used to delete the character immediately before the cursor's position to delete text. The delete key is also used to delete the character at the cursor's position.

The editor also offers options for killing text; the difference between deleting and killing text is that when text is killed, it is placed into a temporary area

and can be retrieved at any time, whereas deleted text is permanently removed. Emacs offers the following commands to kill text:

- You can use the M-Backspace command to kill a word before the present position of the cursor.
- You can use the M-k command to kill everything from the cursor's present position to the ending point of the sentence.
- You can use the M-D command to kill a word after the present position of the cursor.
- You can use the C-k command to kill everything from the cursor's present position to the ending point of the line.

You can also kill text on a large scale by moving the cursor to the beginning of the section you want to be killed, pressing C-spacebar or C-@, and placing the cursor at the end of the section. Then press C-w, and all the indicated text is killed.

Popular Linux Commands

There are far too many Linux commands to discuss in any detail but we have covered some of the most common. If you want to know what any Linux command does, simply access their manual page using the syntax below and the name of the command:

```
$ man command-name
```

The table below shows you many more commands, with their syntax and a description of what they do:

Command	Syntax	Description
adduser/ addgroup	$ sudo adduser $ sudo addgroup	These two commands add users or groups respectively to the system
agetty	$ agetty -L 9600 ttyS1 vt100	A program managing virtual and physical terminals. Invoked by init, it opens a tty port when a connection is made
alias	$ alias home='cd	Creates a shortcut or alias to a specific Linux command

		on the user's system
apropos	$ apropos adduser	Searches and displays a command or program's man page description
apt	$ sudo apt update	High-level package manager for Ubuntu and Debian systems
apt-get	$ sudo apt-get update	Used to install, remove and upgrade software packages and upgrade the operating system
aptitude	$ sudo aptitude update	Text-based GNU/Linux package

		manager system used for installing or removing packages
arch	$ arch	Displays the machine hardware or architecture name
arp	$ sudo arp-scan --	ARP maps IP network addresses to MAC addresses – this command all live hosts on a specified network
at	$ sudo echo	Schedules tasks to be executed at a future specified time
atq	$ atq	Used for

		viewing jobs queued in the at command
atrm	$ atrm (job number)	Used for deleting jobs from the at command queue by their job number
awk	$ awk '	Program created for processing text and as a reporting and data extraction tool
basenam e	$ basename bin/ findhosts.sh	Prints a file name from stripped of directories in the absolute path
bc	$ echo 20.05 + 15.00 | bc	Powerful, recuses CLI

		calculator language – see example under syntax
bzip2	$ bzip2 -z filename #Compress $ bzip2 -d filename.bz2 #Decompress	Compresses or decompresses specified files
cal	$ cal	Prints a calendar
cat	$ cat file.txt	View file contents, concatenate files or data and display it
chgrp	$ chgrp (new) (old)	Changes a file's group ownership. New name is provided first with the

		existing one second
chmod	$ chmod	Changes or updates access permissions for specified files
chown	$ chown	Changes or updates group and user ownership of directories or files
cksum	$ cksum	Displays an input file's byte count and CRC checksum
clear	$ clear	Clears terminal screen
cmp	$ cmp file1 file2	Compares two files byte-by-byte
comm	$ comm file1	Compares

	file2	sorted files on a line-by-line basis
cp	$ cp	Copies directories and files from place to another – locations must be specified
date	$ date	Displays and sets system time and date
df	$ df -h	Displays the disk space usage on a specified file system
diff	$ diff file1 file2	Compares specified files line-by-line and can also find and locate differences

		between directories
dir	$ dir	Similar to ls command, it lists directory contents
dmidecode	$ sudo dmidecode	Retrieves information about a Linux system's hardware
du	$ du	Shows how files use disk space in a directory and related sub-directories
echo	$ echo	Takes a specified line of text and prints it
eject	$ eject	Ejects DVD, CD ROM, and other

		removeable media
env	$ env	Lists and sets current environment variables
exit	$ exit	Exits the shell
expr	$ expr	Calculates a specified expression
factor	$ factor	Shows a specified number's prime factors
find	$ find	Searches a directory and sub-directories for specified files using attributes like users, permissions, data, file type size, etc.

free	$ free	Shows the systems memory use. Add -h to the command to show the info in human readable format
grep	$ grep	Searches files for specified patterns and displays those lines with the pattern
groups	$ groups	Displays the groups a specified user belongs to
gzip	$ gzip	Compresses a specified file and replaces it with a file with a .gz extension

gunzip	$ gunzip	Expands/ restores files compressed using gzip
head	$ (file or stdin) | head	Displays the first 10 lines of the file or stdin specified
history	$ history	Shows commands or retrieves info about commands a user has already executed
hostname	$ hostname	Prints or sets a Linux system hostname
hostnamectl	$ hostnamectl	Takes control of the system hostname in system and modifies or

		prints the hostname and related settings
hwclock	$ sudo hwclock	Manages the hardware clock and reads or sets it
hwinfo	$ hwinfo	Looks into a system to see what the hardware is
id	$ id	Displays group and user information about current or specified username
ifconfig	$ ifconfig	Configures, views and controls the network interfaces for specified Linux systems

ionice	$ ionice	Sets/views process I/O scheduling class and specified process priority
iostat	$ iostat	Shows input/ output and CPU stats for partitions and devices and produces reports on how to update configurations for input/ output balancing
ip	$ sudo ip	Displays/ manages devices, routing, tunnels and policy routing

iptables	$ sudo iptables	Manages incoming/ outgoing traffic using configurable table rules
iw	$ iw list	Manages wireless devices and configurations
iwlist	$ iwlist	Shows detailed info from specified wireless interfaces
kill	$ kill	Uses a specified process's PID to kill it
killall	$ killall (process name)	Kills a specified process using its name
kmod	$ kmod	Manages kernel modules

last	$ last	Shows the last users logged in
ln	$ ln -s	Uses -s flag to create soft links between files
locate	$ locate (file name)	Finds a file by its specified name
login	$ sudo login	Creates a new system session
ls	$ ls	Lists a directory's contents and using the -1 flag will create a long listing
ishw	$ sudo ishw	Provides basic details on the machine's hardware configuration – use superuser privileges

		to get more detailed info
iscpu	$ iscpu	Displays info about the system architecture
lsof	$ lsof	Shows details of files a process has opened
lsusb	$ lsusb	Shows information about system USB buses and any connected devices
man	$ man (command)	Shows specified command or program manual pages
mkdir	$mkdir	Creates one or more directories

more	$ more (file name)	Displays long files one screen at a time
mv	$ mv	Renames directories or files or moves them to specified locations in the structure
nc/ netcat	$ nc/netcat	Performs UDP, TCP or UNIX-domain socket operations
netstat	$ netstat	Shows useful info about the networking subsystem
nice	$ nice	Shows or changes nice value of a specified running program.

		Adjusted niceness must be specified otherwise current niceness is displayed
nmap	$ nmap	Open-source network scanner
nproc	$ nproc	Displays available processing unit number for current processes
passwd	$ passwd	Creates or updates passwords for specified accounts and can also change validity period.
pidof	$ pidof	Shows a

		running command or program's process ID
ping	$ ping	Determines connectivity between network or internet hosts
ps	$ ps	Displays info about active running system processes
pstree	$ pstree	Shows running processes in tree format, rooted at init or PID
pwd	$ pwd	Shows the current or working directory's name

reboot	$ reboot	Used to stop, reboot or turn off a system
rename	$ rename	Can rename multiple files at a time, for example renaming all files with .html extension to .php extension
rm	$ rm (file/ directory name)	Removes specified directories or files
rmdir	$ rmdir (directory name)	Deletes or removes empty directories
scp	$ scp	Allows files to be securely copied between network hosts
shutdow	$ shutdown	Schedules a

n		time to stop, power off or reboot the system/ machine
sleep	$ sleep	Delays or pauses a command execution for a specified time period
sort	$ sort (file name)	Sorts text lines in a specified file or stdin
split	$ split	Splits specified files into smaller bits
ssh	$ ssh	Accesses and runs commands on specified remote machines using an

		encrypted and secure communicatio n over insecure networks
stat	$ stat	Shows the status of a specified file or file system
su	$ su	Switches from one user ID to another or to root in a login session. If no username is specified, the default is root
sudo	$ sudo	Allows system user to run commands as another user or root
sum	$ sum	Displays block counts and

		checksum for all specified files
tac	$ tac	Concatenates specified files and displays them in reverse
tail	$ tail	Displays last 10 lines of specified files to standard output
talk	$ talk person (login name) $ talk ‘user@host’	Talks to other network or system users. Login name is used to talk to a person on the same machine and user@host to talk to a user on a different machine
tar	$ tar	Powerful file

		archiving utility
tree	$ tree	Cross-platform command-line program for recursively displaying or listing directory contents in tree format
time	$ time	Runs a program and provides a summary of the system resource used
top	$ top	Shows all Linux system process CPU and memory usage
touch	$ touch	Changes the timestamp of

		a file or creates files
uname	$ uname	Shows system info, such as version, release date, hostname kernel name, operating system and so on. The -a flag is used to display all information
uptime	$ uptime	Displays length of system running time, how many users are logged on and the load averages
users	$ users	Displays names of current users

vim/vi	$ vim file	Text editor used for editing program and text files
w	$ w	Shows load averages, uptime and info about current users, along with their processes, etc.
wall	$ wall (message)	Sends a specified message to all system users
watch	$ watch	Repeatedly runs a specified program while showing the program output. Can also watch file

		or directory changes
wc	$ wc (filename)	Displays the word, newline and byte counts for specified files and totals for multiple files
wget	$ wget	Downloads web files non interactively
whatis	$ whatis (command)	Displays short man page descriptions of specified commands
which	$ which	Shows absolute paths for specified files that may be executed in the current environment

who	$ who	Shows info about current logged-in users
whereis	$ whereis (command)	Shows manual, source and binary files for specified commands
yes	$ yes (string)	Repeatedly displays a specified string until killed or terminated
zip	$ zip	Packages and compresses archive files

Chapter Eight

Linux Shell Scripting

The command-line tools are good for solving computing problems, but they do not make you the master artist in the world of Linux. The shell makes you the master of Linux. You should know how the shell works and how you can write different types of scripts in a Linux shell. By learning the command line and Linux shell, you will be able to carry out a number of tasks by itself.

A shell script, in the simplest terms, is a file that contains a series of commands. The shell will read the file and carry out the commands as though they have been entered on the command line. The shell emerges out as distinctive in that it is a robust command-line interface system as well as a scripting language interpreter. Most of the things that you can do on the command line can also be done inside shell scripts. Most of the things that you can do in shell scripts can also be done on the command line in Linux.

We now have covered most of the shell scripting features; however, we have focused on the ones that are more often used on the command line. The shell gives us many features when we are writing programs.

Writing the Shell Script

Shell scripts are just like ordinary text files which is why you need a text editor to write the scripts. The best text editors will provide you syntax highlighting, which allows you to see a color-coded view of different script elements. Syntax highlighting will help you spot different kinds of common errors and use kate or vim to write shell scripts. The system is a bit fussy about blocking old text files to run as a program. All this happens for a good reason. You need to set up permissions of the script file to allow the execution. You should also put the script in a place where you can easily find it because the shell searches the file and executes it.

The Format of Linux Shell Scripts

I will now enter the writing phase and show you how you can write a shell script. The language of the script is simple. See the script as under:

```
[gha@localhost ~]$ echo 'I am learning Linux'
I am learning Linux
```

This is shell scripting. Now I will add a comment to the same script.

```
[gha@localhost ~]$ echo 'I am learning Linux' # I have added a comment to the code.
I am learning Linux
```

Now you have to make the script executable. See the following:

```
[gha@localhost ~]$ ls -1 Linux_learning
[gha@localhost ~]$ chmod 755 Linux_learning
```

The text Linux_learning is the name of the file in which I had saved the script. The chmod command is used to make the script executable in Linux.

Displaying Text

Most of the shell commands tend to produce a specific output, which is displayed on your console monitor where you are running the script. You may need to add text messages to help out the script user know what is going to happen inside the script. You can do this by using the echo command I have talked about in the past section. The echo command displays simple text strings. There are different techniques to write a script with the echo command. I will explore all of them.

```
[gha@localhost ~]$ echo I am learning Linux shell scripting
I am learning Linux shell scripting
```

The most remarkable thing is that you do not have to use quotes to enclose the string text, as is the case with other programming languages. However, sometimes it becomes necessary to use quotes. Either you need single or double quotes to display the text strings.

```
[gha@localhost ~]$ echo ‘I am learning Linux shell scripting.’
I am learning Linux shell scripting
```

In the next example, using quotes is going to be inevitable.

```
[gha@localhost ~]$ echo 'Adam says, "I am learning Linux shell scripting."'
Adam says, “I am learning Linux shell scripting."
```

You can see that when you have to add a text that is in the form of direct

speech, you will have to add quotation marks to the text.

The if-then Statement

Many Linux shell scripting programs require some logic flow control between different commands inside the script, which means that the shell tends to execute different commands. In addition, it keeps the ability to execute different other commands that permit the script to loop through the commands based on the result of different other commands. We refer to them as structured commands. They are also known as conditionals.

Structured commands allow you to shift the flow of the operations of a program. Some commands are executed, while others are skipped when they are caught up in certain conditions. In other programming languages, the object after an if statement is the equation that the system evaluates for True or False grounds. The bash shell if statement does not work that way. Instead, it runs the command that is defined on the if line. If this command's exit status is considered zero, the commands that are listed under the then section tend to execute. If this command's exit status is something else, the then commands are put on hold, and the bash shell jumps over to the next command in the script.

```
$ cat testing5
#!/bin/bash
# I am now testing the if statement for bash shell
if date
then
echo "The structured command has completely worked"
fi

$bash -f main.sh
Tue Jan 26 05:02:54 UTC 2021
The structured command has completely worked
```

The script has used the date command on the if line. If this command is executed successfully, the echo statement must display the text string. When you are running the script from the command line, you will be getting the same results as above. The shell will execute the date command. Since the exit status was zero, it will also execute the echo statement that is listed in the then section. In the next example, I will test a bad command to see how it works.

```
$ cat testing2
#!/bin/bash
```

```
# It is time to run a test on a bad command
if hgjdksllh
then
echo "it is not going to work"
fi
echo "I cannot run the command because you’re out of the if statement."

$bash -f main.sh
I cannot run the command because you’re out of the if statement
```

Since the above command was a bad command, it produced an exit status that is at the moment a non-zero. The bash shell will skip the echo statement in the then section.

The if-then-else Statement

The if-then statement provides one option to determine whether the command is successful. If a non-zero exit status code is returned, the bash shell goes to the next script command. In a situation like this, it would best if we had another set of commands we could execute, and that is where the if-then-else statement comes in.

If the if-statement command produces the exit status zero code, the commands in the then part of the statement are executed. If a non-zero exit status code is returned, the commands in the else part of the statement are executed.

```
$ cat testing4
#!/bin/bash
# I am now going to test the else section
testinguser=thisisabadtest
if grep $testinguser /etc/passwd
then
echo The files for user $testinguser are:
ls -a /home/$testinguser/.b*
else
echo "The user name $testinguser cannot be found on the system"
fi
$ ./test4
The user name thisisabadtest cannot be found on the system
$

$bash -f main.sh
The user name thisisabadtest cannot be found on the system
```

On occasion, you may need to check things in the script code. Rather than separate if-then statements, a different version of the else statement can be used. This is called the elif, and this continues the else statement with a

subsequent if-then statement. It provides a different evaluation command, which is much like the first if statement. If a zero exit status code is returned, the commands in the second then statement are executed by the bash shell.

Advanced if-then Features

There is a double parentheses command in Linux that allows you to incorporate some advanced mathematical formulas when you are making comparisons. The test command permits simple operations in comparison. However, the double parentheses command offers more mathematical symbols that a number of programmers from different other languages are using.

The expression term may be any kind of mathematical assignment or expression. Besides the standard operators that the test command uses, additional operators are available for use in the double parentheses command.

You also can use the double bracket command for advanced features to do string comparisons. The double bracket command format uses the standard string comparison that we have used in the test command. Where it differs is by the option of pattern matching. You will be able to define a regular expression that you need to match against the string value.

```
$ cat testing2
#!/bin/bash
# I will now be using pattern matching
if [[ $MyUSER == r* ]]
then
echo "Hello $MyUSER I know you from the days when you used to work in a restaurant in Europe."
else
echo "Sorry, I have never seen you before. I don’t know you."
fi
$bash -f main.sh
```

Sorry, I have never seen you before. I don’t know you.

More often, you may find yourself attempting to evaluate the value of some variables in a bid to find out a specific value inside a set of possible values. In this typical scenario, you may end up writing a lengthy if-then-else statement such as the following:

```
$ cat testing555
#!/bin/bash
# I am now looking for some possible values
```

```
if [ $MyUSER = "rich" ]
then
echo "Welcome $MyUSER to the heavens on the earth."
```

echo "Please enjoy your visit in plush green fields and lush green mountains. I wonder if you want to take a bath in the river beneath the valley."

```
elif [ $MyUSER = Jason ]
then
echo "Welcome $MyUSER to the lakes of Madora."
```

echo "Please enjoy your stay in the shining waters of Madora lakes. I hope you enjoy the resorts, the sun, the breeze, the food, and the drinks."

```
elif [ $MyUSER = Simra ]
then
echo "Welcome $MyUSER to the lakes of Madora."
```

echo "Please enjoy your stay in the shining waters of Madora lakes. I hope you enjoy the resorts, the sun, the breeze, the food, and the drinks."

```
elif [ $MyUSER = Linda ]
then
echo "Don’t forget to logout when you’re done"
else
echo "Sorry, you’re not allowed here"
fi
```

The for Command

Iterations through a series of commands is normal practice when it comes to programming. Many a time, you may have to repeat different commands. Often you need to repeat a bunch of commands until the program meets a specified condition like processing files inside a directory or in all the lines inside a text file.

The bash shell offers the for command to develop a loop that iterates through a set of values. Each iteration tends to perform a set of commands by using a single value inside the series. One of the most basic uses of the for command is iteration through a list of values that are defined inside the for command itself.

```
v=$ cat testing111
#!/bin/bash
# This is the basic for command in Linux
```

for mytest in Alabama Texas Alaska Virginia Arizona Ohio Arkansas South Dakota California North Dakota Colorado New Jersey Florida Michigan

Georgia

```
do
```

echo I am planning to continue my travel circle across the United States in the summer. The next state I will visit is $mytest

```
done
```
$bash -f main.sh

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Alabama

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Texas

And so on until the last line:

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Georgia

Each time the for command iterates through the list of objects, it assigns the mytest variable to the next item in the list. The $mytest variable is used like other script variables in the for command statements. After the final iteration, the $test variable remains valid throughout the shell script's remaining part. It will retain the final iteration value unless you change it.

```
v=$ cat testing111
#!/bin/bash
# This is the basic for command in Linux
```

for mytest in Alabama Texas Alaska Virginia Arizona Ohio Arkansas South Dakota California North Dakota Colorado New Jersey Florida Michigan Georgia

```
do
```

echo I am planning to continue my travel circle across the United States in the summer. The next state I will visit is $mytest

```
done
echo "The last US state I traveled to was $mytest"
mytest=Connecticut
echo "Now I am going to visit $mytest"
```

$bash -f main.sh

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Alabama

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Texas

And so on, until the last line:

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Georgia

The last US state I traveled to was Georgia

Now I am going to visit Connecticut

You can see that the $mytest variable retained its original value and permitted us to change it outside the for loop. Let's see another for loop example:

```
v=$ cat testing111
#!/bin/bash
# This is the basic for command in Linux
```

for mytest in pumpkin potato tomato spinach ginger garlic cauliflower cabbage pepper beet ladyfinger broccoli

```
do
```

echo I am planning to make a shift to the cooking schedule as the summer sets in. I have bought lots of vegetables. However, today I plan to cook $mytest.

```
done
echo "The last vegetable I cooked was $mytest"
mytest=onion
echo "Now I am going to cook $mytest"
```

$bash -f main.sh

I am planning to make a shift to the cooking schedule as the summer sets in. I have bought lots of vegetables. However, today I plan to cook pumpkin.

I am planning to make a shift to the cooking schedule as the summer sets in. I have bought lots of vegetables. However, today I plan to cook potato.

This continues through each of the vegetabes, ending with:

I am planning to make a shift to the cooking schedule as the summer sets in. I have bought lots of vegetables. However, today I plan to cook broccoli.

The last vegetable I cooked was broccoli

Now I am going to cook onion

When you are dealing with the for command, the biggest problem you may encounter is using multi-word values. The for loop assumes that each value ought to be separated by a space. If you have data values to contain spaces, you may run into another problem. You may see that in the example of US states. The bash editor considered South and Dakota as two separate values.

The for command offers us the quotation marks to separate values that have more than one word. You may use double quotes to separate different values in the for command. See the following example.

```
v=$ cat testing111
#!/bin/bash
# This is the basic for command in Linux

for mytest in Alabama Texas "New York" Alaska Virginia Arizona Ohio Arkansas "South Dakota" California "North Dakota" Colorado "New Jersey" Florida Michigan Georgia "North Carolina" "South Carolina" "New Mexico" "Rhode Island" "West Virginia" "New Hampshire" Vermont

do

echo I am planning to continue my travel circle across the United States in the summer. The next state I will visit is $mytest

done
echo "The last US state I traveled to was $mytest"
mytest=Connecticut
echo "Now I am going to visit $mytest"

$bash -f main.sh
```

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Alabama.

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Texas.

Continuing through the States until we get to the last one:

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Vermont

The last US state I traveled to was Vermont

Now I am going to visit Connecticut

Now the for command can distinguish between the single word and multi-word values. Also, the best thing is that when you are using double quotation marks, the shell sheds the quotation marks as a part of the value.

Reading a List through a Variable

Often what happens inside a shell script is that you accumulate lists of values that are stored inside a variable. Then it needs to iterate through a list. You may do this with the help of the for command.

```
$ cat testing111
#!/bin/bash
# This is the basic for command in Linux
```

list="Alabama Texas New York Alaska Virginia Arizona Ohio Arkansas South Dakota California North Dakota Colorado New Jersey Florida Michigan Georgia North Carolina South Carolina New Mexico Rhode Island West Virginia New Hampshire Vermont"

```
list=$list" Idaho"

for state in $list
do
```

echo "I am planning to continue my travel circle across the United States in the summer. The next state I will visit is $state"

```
done
```

$bash -f main.sh

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Alabama

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Texas

Right up to the last state:

I am planning to continue my travel circle across the United States in the

summer. The next state I will visit is Idaho

The $list variable contains the list of values that have been used for iterations. You should take note that the code uses another assignment statement for the concatenation of items to the existing list. Even when you forget to add items in the list, you can add them up by the concatenation method later on. I will add five more items to the list in the next example.

```
$ cat testing111
#!/bin/bash
# This is the basic for command in Linux
list="Alabama Texas Alaska Virginia Arizona Ohio Arkansas California Colorado Florida
Michigan Georgia Vermont Iowa Nebraska Arkansas Kansas Kentucky Tennessee Utah
Maine Missouri Minnesota"
list=$list" Idaho"
list=$list" Oregon"
list=$list" Montana"
list=$list" Delaware"

for state in $list
do
```

echo "I am planning to continue my travel circle across the United States in the summer. The next state I will visit is $state"

```
done
```

$bash -f main.sh

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Alabama

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Texas

This output continues to list the states, right up to the last one:

I am planning to continue my travel circle across the United States in the summer. The next state I will visit is Delaware

There is a way to generate values for custom usage in a list to use the output of a particular command.

```
$ cat testing111
#!/bin/bash
# I will read values from files
myfile="states"
```

```
for state in `cat $myfile`
do
echo "I am planning to continue my travel circle across the United States in the summer. The next state I will visit is $state"
done
$cat states
Alabama
Texas
Alaska
Virginia
Arizona
Ohio
Arkansas
California
Colorado
Florida
Michigan
Georgia
Vermont
Iowa
Nebraska
Arkansas
Kansas
Kentucky
Tennessee
Utah
Maine
Missouri
Minnesota
Idaho
Oregon
Montana
Delaware
```

The while Command

The while command is a cross between the if-then statement and Linux for loop. It allows you to define a command to test a condition and then looping through a set of commands until you reach a zero exit status. It will analyze the test command at the start of each iteration. Upon reaching the non-zero exit status, the while command stops the execution of the set of commands.

```
$ cat test10
#!/bin/bash
# This is a while command test in Linux
var15=20
while [ $var15 -gt 0 ]
do
echo This is $var15 in the while loop.
var15=$[ $var15 - 1 ]
done
```

$bash -f main.sh

```
This is 20 in the while loop.
This is 19 in the while loop.
```

```
This is 18 in the while loop.
The loop continues to the final line:
This is 1 in the while loop.
```

Multiple Test Commands

You can pack up multiple test commands in the while loop.

```
$ cat test1133
#!/bin/bash
# Now I am testing a multicommand while loop in Linux shell
var15=20
while echo $var15
[ $var15 -ge 0 ]
do
echo "This operation is being conducted inside the while loop."
var15=$[ $var15 - 1 ]
done
```

$bash -f main.sh

```
20
This operation is being conducted inside the while loop.
19
This operation is being conducted inside the while loop.
18
This operation is being conducted inside the while loop.
17
This will continue until it reaches the last number:
This operation is being conducted inside the while loop.
0
This operation is being conducted inside the while loop.
-1
```

The first test command displays the present value of var15 variable. The second command will use the test command to know the value of var15 variable. The echo statement returns a simple message within the loop that indicates that the loop has runs its course.

The until Command

The until command works opposite to the while command. It requires a simple test command is specified, which will produce a non-zero exit status, at which point the bash shell will execute the while loop commands.

However, similar to the while command, the until command statement can also contain multiple test commands, and the exit status of the final command is the determining factor.

```
$ cat test1222
#!/bin/bash
# In this code block I will be using the until command
var15=100
until [ $var15 -eq 0 ]
do
echo You are seeing the operations of the until command. The next digit in the loop is
$var15
var15=$[ $var15 - 5 ]
done
```

$bash -f main.sh

You are seeing the operations of the until command. The next digit in the loop is 100

You are seeing the operations of the until command. The next digit in the loop is 95

You are seeing the operations of the until command. The next digit in the loop is 90

And continuing, until it reaches the final digit:

You are seeing the operations of the until command. The next digit in the loop is 5

The example has successfully tested the var15 variable in order to determine when the loop will stop - when the value of the variable drops down to zero, the until command stops right away.

```
$ cat test1222
#!/bin/bash
# In this code block, I will be using the until command
var15=100

until echo $var15
        [ $var15 -eq 0 ]
do
echo You are seeing the operations of the until command. The next digit in the loop is
$var15
var15=$[ $var15 - 5 ]
done
```

$bash -f main.sh

```
100
```

You are seeing the operations of the until command. The next digit in the loop is 100

95

You are seeing the operations of the until command. The next digit in the loop is 95

90

Again, this will continue to the last digit in the loop:

You are seeing the operations of the until command. The next digit in the loop is 5

0

Nesting Loops

A loop statement may use another command inside of the loop, including other commands of the same loop. The process is known as nested looping. You will have an iteration inside another iteration, which will multiply the number of times a command is being run. See an example to get a grasp of how you can nest loops.

```
$ cat test1444
#!/bin/bash
# The following code block shows nesting for loops
for (( x = 1; x <= 7; x++ ))
do
    echo "You are seeing the operations of the until command. The next digit in the loop is
$x:"
    for (( y = 1; y <= 7; y++ ))
    do
        echo " Now you are seeing what is going on inside loop: $y"
    done
done
```

$bash -f main.sh

You are seeing the operations of the until command. The next digit in the loop is 1:

Now you are seeing what is going on inside loop: 1

Now you are seeing what is going on inside loop: 2

And so on, finishing with

Now you are seeing what is going on inside loop: 7

You are seeing the operations of the until command. The next digit in the loop is 2:

Now you are seeing what is going on inside loop: 1

Now you are seeing what is going on inside loop: 2

Again, this will continue until it ends with

Now you are seeing what is going on inside loop: 7

You are seeing the operations of the until command. The next digit in the loop is 3:

Now you are seeing what is going on inside loop: 1

Now you are seeing what is going on inside loop: 2

Ending with:

Now you are seeing what is going on inside loop: 7

You are seeing the operations of the until command. The next digit in the loop is 4:

Now you are seeing what is going on inside loop: 1

Now you are seeing what is going on inside loop: 2

Ending with:

Now you are seeing what is going on inside loop: 7

The entire cycle continues through the loop, until it reaches:

Now you are seeing what is going on inside loop: 7

You can experiment with the numbers in the loop to contract or expand the two loops' sizes. By changing the numbers of the loops, you will be able to change how the loop will behave.

```
$ cat test1444
#!/bin/bash
# The following code block shows nesting for loops
for (( x = 1; x <= 3; x++ ))
```

```
do
    echo "You are seeing the operations of the until command. The next digit in the loop is
$x:"
    for (( y = 1; y <= 5; y++ ))
    do
        echo " Now you are seeing what is going on inside loop: $y"
    done
done
```

$bash -f main.sh

You are seeing the operations of the until command. The next digit in the loop is 1:

Now you are seeing what is going on inside loop: 1

Repeated to:

Now you are seeing what is going on inside loop: 5

You are seeing the operations of the until command. The next digit in the loop is 2:

Now you are seeing what is going on inside loop: 1

Repeated to:

Now you are seeing what is going on inside loop: 5

You are seeing the operations of the until command. The next digit in the loop is 3:

Now you are seeing what is going on inside loop: 1

And finishing on:

Now you are seeing what is going on inside loop: 5

The nested loop iterates through the values for iterations of the outer loop. There is no difference between the do and done commands for two loops. You can create a nested loop by pairing up for loops and while loops.

```
$ cat test15
#!/bin/bash
# This is how we can place a for loop in a while loop
var15=15
while [ $var15 -ge 0 ]
do
```

```
    echo "You are seeing the operations of nesting loops. The next digit in the loop is
$var15:"
    for (( var12 = 1; $var12 < 3; var12++ ))
    do
        var13=$[ $var15 * $var12 ]
        echo " Now you are seeing what is going on inside loop: $var15 * $var12 = $var13"
    done
    var15=$[ $var15 - 1 ]
done
```

$bash -f main.sh

You are seeing the operations of nesting loops. The next digit in the loop is 15:

Now you are seeing what is going on inside loop: 15 * 1 = 15

Now you are seeing what is going on inside loop: 15 * 2 = 30

This will continue down to the final line:

You are seeing the operations of nesting loops. The next digit in the loop is 0:

Now you are seeing what is going on inside loop: 0 * 1 = 0

Now you are seeing what is going on inside loop: 0 * 2 = 0

In the next example, I will pair up until and while loops to test nesting loops' limits. See the following example.

```
$ cat test16
#!/bin/bash
var15=10
until [ $var15 -eq 0 ]
do
echo "You are seeing the operations of the nested until and while loops: $var15"
var12=1
while [ $var12 -lt 5 ]
do
var13=`echo "scale=4; $var15 / $var12" | bc`
echo " This is how the inner section of the nested loop functions: $var15 / $var12 = $var13"
var12=$[ $var12 + 1 ]
done
var15=$[ $var15 - 1 ]
done
```

$bash -f main.sh

You are seeing the operations of the nested until and while loops: 10

This is how the inner section of the nested loop functions: 10 / 1 = 10.0000

This is how the inner section of the nested loop functions: 10 / 2 = 5.0000

This is how the inner section of the nested loop functions: 10 / 3 = 3.3333

This is how the inner section of the nested loop functions: 10 / 4 = 2.5000

This cycle is repeated, until it reaches the final one of:

You are seeing the operations of the nested until and while loops: 1

This is how the inner section of the nested loop functions: 1 / 1 = 1.0000

This is how the inner section of the nested loop functions: 1 / 2 = .5000

This is how the inner section of the nested loop functions: 1 / 3 = .3333

This is how the inner section of the nested loop functions: 1 / 4 = .2500

Loop Control

You might think that once a loop starts, you will be stuck in that until the loop has runs its course. However, there is a way out. You can add to the code a couple of commands that help you control the inside of the loop.

The break command is the simplest way to move out of a loop that is in progress. You may use the break command to exit while and until loops.

```
$ cat test17
#!/bin/bash
for countries in 2 3 4 5 6 7 8 9 10 11 12 13 14
do
if [ $countries -eq 12 ]
then
break
fi
echo "I am going to visit $countries after Covid-19 is over."
done
echo "I have visited all the countries."
```

$bash -f main.sh

I am going to visit 2 countries after Covid-19 is over.

I am going to visit 3 countries after Covid-19 is over.

I am going to visit 4 countries after Covid-19 is over.

I am going to visit 5 countries after Covid-19 is over.

I am going to visit 6 countries after Covid-19 is over.

I am going to visit 7 countries after Covid-19 is over.

I am going to visit 8 countries after Covid-19 is over.

I am going to visit 9 countries after Covid-19 is over.

I am going to visit 10 countries after Covid-19 is over.

I am going to visit 11 countries after Covid-19 is over.

I have visited all the countries.

The for loop has run its course until the use of the break keyword. When the if-then condition was satisfied, the bash shell executed the break command, which put a stopper to the for loop. This technique works equally well for until and while loops.

```
$ cat test18
#!/bin/bash
# breaking out of a while loop
countries=1
while [ $countries -lt 15 ]
do
if [ $countries -eq 8 ]
then
break
fi
echo "I am going to visit $countries after Covid-19 is over."
countries=$[ $countries + 1 ]
done
echo "I have visited all the countries."
```

$bash -f main.sh

I am going to visit 1 countries after Covid-19 is over.

I am going to visit 2 countries after Covid-19 is over.

All the way down to:

I am going to visit 7 countries after Covid-19 is over.

I have visited all the countries.

In the next example, I will use the break statement in the for loop.

```
$ cat test1444
#!/bin/bash
# The following code block shows nesting for loops
for (( x = 1; x <= 4; x++ ))
do
    echo "You are seeing the operations of the until command. The next digit in the loop is
$x:"
    for (( y = 1; y <= 7; y++ ))
    do
    if [ $y -eq 5 ]
    then
        break
    fi
        echo " Now you are seeing what is going on inside loop: $y"
    done
done
```

$bash -f main.sh

You are seeing the operations of the until command. The next digit in the loop is 1:

Now you are seeing what is going on inside loop: 1

Now you are seeing what is going on inside loop: 2

Now you are seeing what is going on inside loop: 3

Now you are seeing what is going on inside loop: 4

This cycle is repeated for four digits and ends with:

You are seeing the operations of the until command. The next digit in the loop is 4:

Now you are seeing what is going on inside loop: 1

Now you are seeing what is going on inside loop: 2

Now you are seeing what is going on inside loop: 3

Now you are seeing what is going on inside loop: 4

You can either redirect the output of the a loop in the shell script or pipe it out for use. You will be needing a processing command to add to the end of the done command.

Loops are a integral part of many programming languages, including Linux and there are three types of looping commands in the bash shell that can be used in scripts.

The first is the for command, which lets you iterate through a list of values in the command line, contained in a variable, or obtained through file globbing (used for file extraction.).

The second is the while command, offering a method of looping by using the command's condition and ordinary commands. In this way, different variable conditions can be tested and, so long as a zero exit status is returned, the while loop will continue iterating through a set of specified commands.

The third is the until command, offering a way of iterating through commands, but based on a command or condition that produces a non-zero exit status. Using this feature, you can set a condition that must be met before the end of the iteration. Loops can be paired up in shell scripts by producing different loop layers. The bash shell provides the continue and break commands that we use to change the normal loop flow process on multiple loop values.

More Basic Shell Scripts

Here are some more simple examples of bash shell scripts that you can try for yourself:

The Hello World Program

Most new programmers start learning their chosen language using the help world program. It is one of the most simple programs that does nothing more than prints a simple string to the standard output, reading "Hello World." In Linux, editors like nano or vim can be used to create the hello-world.sh file and then copy and paste the code below into it:

```
#!/bin/bash
echo "Hello World"
Save the file and exit.
```

This file needs to be made executable with the command below:

```
$ chmod a+x hello-world.sh
```

And then, you can run the file using either of these commands:

```
$ bash hello-world.sh
$ ./hello-world.sh
```

The result is the string you passed to echo being printed on the screen.

How to Print With echo

The echo command will print information from bash and is not unlike the printf function in C. Indeed, they share many of the same options, including re-direction and escape sequences.

Create a new file and call it echo.sh. Make the file executable using the directions in the first example:

```
#!/bin/bash
echo "Printing text"
echo -n "Print text without newline"
echo -e "\nRemoving \t special \t characters\n"
```

Run this script to see what it will do. There are two options in the script: -n is used for newline, and -e tells echo that you have passed a string containing some special characters and it needs extra functionality.

How to Use Comments

Comments are one of the most useful parts of any script and are required for higher-quality codes. They tell the coder and others reading the code what it does but should not be long and rambling – keep them short and to the point. Common practice has comments places inside any code that contains critical logic. The # symbol is used to comment a line, as you can see in the example below:

```
#!/bin/bash

# Add two values
((sum=25+45))

#Print result
echo $sum
```

The script will give us a result of 70. Have a look at how comments are used before each line, except for the first one. The first line is a shebang, and it tells the system the interpreter it should use to run the script.

How to Use Multi-Line Comments

Some coders document their scripts using multi-line comments, and you can see how this is done below:

```
#!/bin/bash
: '
```

This script will calculate

```
the square of 5.
'
((area=5*5))
echo $area
```

Multi-line comments do not use the #; instead, they are enclosed in the' and ' characters.

How to Use While Loops

The while loop is used when you want to run an instruction several times. Have a look at the script below to see how this works:

```
#!/bin/bash
i=0

while [ $i -le 2 ]
do
echo Number: $i
((i++))
done
```

The while loop takes the form you can see below:

```
while [ condition ]
do
commands 1
commands n
done
```

You must include the spaces surrounding the brackets – this is not optional; it is mandatory. Missing it out will result in errors.

How to Use the For Loop

Another popular and much user construct is the for loop. This is used for efficient iteration over a set of code, and you can see a simple example here:

```
#!/bin/bash

for (( counter=1; counter<=10; counter++ ))
do
echo -n "$counter "
done

printf "\n"
```

Create a file and name it for.sh, save this code, and make it executable. Then run it, and you should see the numbers from 1 to 10 printed on your screen.

How to Receive User Input

Implementing user interaction in bash shell scripts is critical and requires that you get user input. The example below shows you how to get user input in a shell program:

```
#!/bin/bash

echo -n "Enter anything:"
read anything

echo "You Entered: $anything."
```

In this script, a variable name follows the read construct, and this is how we get user input. The variable is used to store the user input and the $ symbol used to access it.

How to Use the If Statement

The if statement is the most common of the conditional constructs using in shell scripting, and they take this format:

```
if CONDITION
then
STATEMENTS
fi
```

The statements will be executed if the CONDITION is true. The fi keyword marks the end of the statement, and you can see how it all works in the example below:

```
#!/bin/bash

echo -n "Enter a number: "
read num

if [[ $num -gt 10 ]]
then
echo "Number is greater than 10."
fi
```

You will only get an output from this if the given input number is more than 10. The -gt option indicates "greater than" while -lt, if used, will indicate "less than." -le is for "less than equal," and -ge is for "greater than equal." You must include the [[]] – they are not optional.

How to Get More Control Using If Else

You can get more control over the logic in the script by combining an else and if construct. Have a look at the example:

```
#!/bin/bash

read n
if [ $n -lt 10 ];
then
echo "It is a one-digit number."
else
echo "It is a two-digit number."
fi
```

The else section must go after the action bit of the if statement before fi, which closes the statement.

How to Use the AND Operator

We can use the AND operator to check whether several conditions have been satisfied. Every part that the operator separates has to be true; otherwise, the AND statement will only return false. Have a look at the script below to see how the operator works:

```
#!/bin/bash

echo -n "Enter Number:"
read num

if [[ ( $num -lt 10 ) && ( $num%2 -eq 0 ) ]]; then
echo "Even Number"
else
echo "Odd Number"
fi
```

We use the double-ampersand (&&) to denote the AND operator.

How to Use the OR Operator

Another important construct is the OR operator. Using it helps us to implement robust and complex logic in a shell script. Unlike the AND operator, any statement with an OR operator will return true when only one operand is true. False is returned when the operand on each side of the OR operator is false. Here's an example:

```
#!/bin/bash

echo -n "Enter any number:"
read n

if [[ ( $n -eq 15 || $n -eq 45 ) ]]
```

```
then
echo "You won"
else
echo "You lost!"
fi
```

This is a simple example showing the OR operator working in a Linux shell script. The user will only be declared a winner when the number 15 or 45 is input.

We use the || sign to denote the OR operator.

How to Use Elif

The elif statement means "else-if" and is used to implement some chain logic. Have a look at this example to see how it works:

```
#!/bin/bash

echo -n "Enter a number: "
read num

if [[ $num -gt 10 ]]
then
echo "Number is greater than 10."
elif [[ $num -eq 10 ]]
then
echo "Number is equal to 10."
else
echo "Number is less than 10."
fi
```

This is a self-explanatory example, so I won't go into too much detail. You can play around and change the variable names or their values to see how they function.

How to Use the Switch Construct

This is another very powerful construct in Linux scripts, and you can use it when you need to use nested conditions, but you don't want to write complicated of-else-elif chains. Here is an example of how it works.

```
#!/bin/bash

echo -n "Enter a number: "
read num

case $num in
100)
echo "Hundred!!" ;;
200)
echo "Double Hundred!!" ;;
```

```
*)
echo "Neither 100 nor 200" ;;
esac
```

The conditions go in between two keywords – case and esac. And we use *) to match all the inputs that are not 100 or 200.

How to Use Command Line Arguments

There are several reasons why you might want to get an argument straight from the command shell, and the next example demonstrates how you do this using bash shell:

```
#!/bin/bash
echo "Total arguments : $#"
echo "First Argument = $1"
echo "Second Argument = $2"
```

Run the script with an extra two parameters following the name. The script is saved as test.sh, and you can see the calling procedure here:

```
$ ./test.sh Hey Howdy
```

$1 accesses the first argument while $2 does the second and so on. We use $# to get the total argument number.

How to Use Names to Get Arguments

The next example demonstrates how you can use their names to get the command line arguments:

```
#!/bin/bash

for arg in "$@"
do
index=$(echo $arg | cut -f1 -d=)
val=$(echo $arg | cut -f2 -d=)
case $index in
X) x=$val;;
Y) y=$val;;
*)
esac
done
((result=x+y))
echo "X+Y=$result"
```

Save and call this script test.sh and call it like this:

```
$ ./test.sh X=44 Y=100
```

You should see a return of X+Y=144. We store the arguments in 'S@' and then use the cut command in Linux to get them.

How to Concatenate Strings

Many modern shell scripts use string processing and one of the most common methods is string concatenation. Bash makes this easy and precise and the example below shows you how it works:

```
#!/bin/bash

string1="Ubuntu"
string2="Pit"
string=$string1$string2
echo "$string is a fantastic Linux resource for beginners."
```

Running this should give you an output of "UbuntuPit is a great Linux Resource for Beginners.:

How to Slice Strings

Most programming languages provide ready-made functions to help you cut parts of a string. Bash doesn't but the example below shows how to do it with a parameter expansion:

```
#!/bin/bash
Str="Learn the Bash Commands from UbuntuPit"
subStr=${Str:0:20}
echo $subStr
```

Running this script should result in "Learn the Bash Commands" as an output. The parameter expansion is the form of ${VAR_NAME:S:L} with S denoting the start of the slice and L indicating how long the slice should be.

How to Use Cut to Extract Substrings

Linux has a command called cut that you can use in your scripts to cut a part out of the string – this is the substring and the next example demonstrates how this works:

```
#!/bin/bash
Str="Learn the Bash Commands from UbuntuPit"
#subStr=${Str:0:20}

subStr=$(echo $Str| cut -d ' ' -f 1-3)
echo $subStr
```

How to Add Two Values

Arithmetic operations are easy to do in a Linux shell script. You can see from the example below how two numbers are taken as an input and added together:

```
#!/bin/bash
echo -n "Enter the first number:"
read x
echo -n "Enter the second number:"
read y
(( sum=x+y ))
echo "The result of the addition=$sum"
```

You can see from this that it is quite straightforward to add two numbers in the bash shell.

How to Add Multiple Values

If you want to get multiple user inputs in your script and add them, you would use loops. The next example shows how this is done:

```
#!/bin/bash
sum=0
for (( counter=1; counter<5; counter++ ))
do
echo -n "Enter the Number:"
read n
(( sum+=n ))
#echo -n "$counter "
done
printf "\n"
echo "Result is: $sum"
```

If you do not include the (()), you wont get an addition operation. Instead it will be concatenation so make sure you double check your script before executing it.

Bash Functions

All computer programming languages rely on functions and the Linux Shell script is no exception. Functions allow you to create blocks of code that you need to use frequently, rather than writing the same code repeatedly. In the next demonstration, you can see how these functions work in bash scripts:

```
#!/bin/bash
function Add()
{
echo -n "Enter a Number: "
read x
echo -n "Enter another Number: "
read y
echo "Addition is: $(( x+y ))"
}

Add
```

What we did here was added two numbers but, unlike the previous addition example, this time we used a function named Add. Whenever you need to do an addition calculation again, you can simply call the function.

Functions with Return Values

One of the best things about functions is that you can use them to pass data between functions. This is useful in lots of different scenarios, so take a look at the next example to see how it works:

```
#!/bin/bash

function Greet() {

str="Hello $name, what brings you to UbuntuPit.com?"
echo $str
}

echo "-> what's your name?"
read name

val=$(Greet)
echo -e "-> $val"
```

The output will show data coming from the function called Greet.

How to Use Bash Scripts to Create Directories

One way that developers can increase their productivity is to use shell scripts to run system commands. The next example is a simple one demonstrating how directories are created in a shell script:

```
#!/bin/bash
echo -n "Enter directory name ->"
read newdir
cmd="mkdir $newdir"
eval $cmd
```

Take a close look at this script and you will see that it does nothing more than call a shell command named mkdir and passing the directory name to it. The result should be a new directory created in your filesystem. The execution command can also be passed inside backticks (') as you can see below:

```
`mkdir $newdir`
```

How to Create a Directory After Checking For Existence

The program we created above won't work if there is a folder named the same

in the current working directory. Instead, we need to check if there is a folder called $dir first and, if there isn't we can create one. Here's how it's done:

```
#!/bin/bash
echo -n "Enter the directory name ->"
read dir
if [ -d "$dir" ]
then
echo "The directory exists"
else
`mkdir $dir`
echo "Directory created"
fi
```

If you want to improve your bash scripting skills, write the above program using eval.

How to Read Files

Bash scripts offer an effective way of reading files. The example below demonstrates how to use shell scripts to read a file. First create a new file and name it editors.txt, make it executable and add the following:

1. Vim
2. Emacs
3. ed
4. nano
5. Code

This code will result in the five lines being printed on your screen.

```
#!/bin/bash
file='editors.txt'
while read line; do
echo $line
done < $file
```

How to Delete Files

In the next program, we will look at how files are deleted within a shell script. First, the program asks the user for input in the form of the filename. If it exists, the file will be deleted. In this example, the rm command is used for the deletion:

```
#!/bin/bash
```

```
echo -n "Enter the filename ->"
read name
rm -i $name
```

Where it asks for the filename, type in editors.txt and, when you are asked to confirm it, press the Y key. The file should be deleted.

How to Append to Files

The next shell example shows you how to use bash scripts to append data to one of your filesystem files. This will add an extra line into the editors.txt file:

```
#!/bin/bash
echo "Before you append the file"
cat editors.txt
echo "6. NotePad++" >> editors.txt
echo "After you append the file"
cat editors.txt
```

By now, you should have noticed that standard terminal commands are being used from the bash scripts.

How to Test For File Existence

In the next example, we can see how to use bash programs to find out if a file exists:

```
#!/bin/bash
filename=$1
if [ -f "$filename" ]; then
echo "File exists"
else
echo "File does not exist"
fi
```

The argument is the filename and it is passed directly from the command-line.

How to Send Mail from Shell Scripts

Sending emails from a bash script is quite simple to do. In the example below, you can see one way to do this:

```
#!/bin/bash
recipient=" admin@example.com"
subject=" Greetings"
message=" Welcome to UbuntuPit"
`mail -s $subject $recipient <<< $message`
```

This will send an email to the specified recipient with the specified subject

line and message.

How to Parse the Date and Time

In the next script example, you can see how to use scripts to handle time and date. We use the data command to get the required information and the script will do the necessary parsing:

```
#!/bin/bash
year=`date +%Y`
month=`date +%m`
day=`date +%d`
hour=`date +%H`
minute=`date +%M`
second=`date +%S`
echo `date`
echo "Current Date is: $day-$month-$year"
echo "Current Time is: $hour:$minute:$second"
```

Save and run the program to see how it all works and also run the date command form your own terminal.

Run this program to see how it works. Also, try running the date command from your terminal.

How to Use the Sleep Command

You can use the sleep command to make your shell script pause in between each instruction. This can be useful for many things, including when you want to perform system-level jobs. In the next example, the sleep command can be seen working in a shell script:

```
#!/bin/bash
echo "How long should you wait?"
read time
sleep $time
echo "Waited for $time seconds!"
```

The program stops the execution of the final instruction for the specified time – that time is provided by the user.

How to Use the Wait Command

We can use the wait command to pause bash script system processes. The example below demonstrates in detail how this all works in a bash script:

```
#!/bin/bash
echo "Testing the wait command"
sleep 5 &
pid=$!
```

```
kill $pid
wait $pid
echo $pid was terminated.
```

Save and run the program and see for yourself how it all works.

How to Display the Last Updated File

On occasion, you may need to locate the last file that was updated – this may be needed for specific operations. Below you can see a simple program that shows you how to use the awk command in bash to do this. It will list one of two things – the last file updates or the last file created in the current working directory.

```
#!/bin/bash

ls -lrt | grep ^- | awk 'END{print $NF}'
```

Copy this code into a file and run it to see how it all works.

How to Add Batch Extensions

In the program below, we are applying a custom extension to every file inside a specified directory. For the purposes of this demonstration, create a brand new directory and put a few files in it. In my folder, I have five files. Each is called test followed by a number between 0 and 4. For the sake of this demonstration the script will add .UP to the end of each file – you can add whatever extension you want, just make sure you change the script accordingly.

```
#!/bin/bash
dir=$1
for file in `ls $1/*`
do
mv $file $file.UP
done
```

One important note – you should not try running this script from a regular directory, only from a test one. You must also provide a command-line argument containing the directory name for the files – the period (.) should be used for the current working directory.

How to Print the Number of Files or Directories

The script below will find how many folders or files are in a specified directory, using the find command provided in Linux to do it. The name of the directory where you are looking for the files should be passed from the

command-line.

```
#!/bin/bash

if [ -d "$@" ]; then
echo "Files found: $(find "$@" -type f | wc -l)"
echo "Folders found: $(find "$@" -type d | wc -l)"
else
echo "[ERROR] Please retry with another folder."
exit 1
fi
```

If the directory you specify is not available you will be asked to try again with a different one. The same message will appear if you do not have permission to access the specified directory.

How to Clean Log Files

In the next demonstration, you can see how shell scripts can be used easily in real life. We are going to delete every log file stored in the directory called /var/log. If you want to clean up different logs, simply change the variable holding the directory:

```
#!/bin/bash
LOG_DIR=/var/log
cd $LOG_DIR

cat /dev/null > messages
cat /dev/null > wtmp
echo "Logs cleaned up."
```

Important – this shell script must be run as root.

Using Bash to Back Up Your Script

Shell scripts provide a strong, effective way of backing up directories and files. In the next example, you can see how to back up every file and directory that was modified in the preceding 24 hours. The Linux find command is used for this.

```
#!/bin/bash

BACKUPFILE=backup-$(date +%m-%d-%Y)
archive=${1:-$BACKUPFILE}

find . -mtime -1 -type f -print0 | xargs -0 tar rvf "$archive.tar"
echo "Directory $PWD backed up in archive file \"$archive.tar.gz\"."
exit 0
```

The file and directory names will be printed once the backup is successfully completed.

How To See If a User is Root

In the next example, you can see how a bash script is used to see if a user is root:

```
#!/bin/bash
ROOT_UID=0

if [ "$UID" -eq "$ROOT_UID" ]
then
echo "You are root."
else
echo "You are not root"
fi
exit 0
```

The output will depend entirely on which user is running the script. Root users are matched by the $UID.

How to Remove Duplicate Lines From a File

It can take a lot of time to process files and it can reduce admin productivity severely. One of the most time-consuming tasks is trying to find duplicate lines in your files but a short, simple shell script can save the day:

```
#! /bin/sh

echo -n "Enter the Filename-> "
read filename
if [ -f "$filename" ]; then
sort $filename | uniq | tee sorted.txt
else
echo "No $filename in $pwd...try again"
fi
exit 0
```

This script will look through every line of the specified file and will remove any duplicate. Those lines are then put into a new file, leaving the original file as it is.

How to do System Maintenance

If you want to make life a little easier for yourself, you can use a small shell script to upgrade your system, rather than having to do it all manually. Here's an example of how this is done:

```
#!/bin/bash
```

```
echo -e "\n$(date "+%d-%m-%Y --- %T") --- Starting work\n"

apt-get update
apt-get -y upgrade

apt-get -y autoremove
apt-get autoclean

echo -e "\n$(date "+%T") \t Script Terminated"
```

This script will also remove old packages that you do not need anymore. Make sure to run it using sudo – if you don't, it won't work.

Shell scripting in Linux isn’t as hard as you might think it is, and your scripts can be as diverse as you want. There is no limit in what they can or cannot do and, if you are new to this, new to using Linux and the bash script, take the time to learn these fundamental scripts – they will form the basis of just about everything you want to do. Once you have learned them, take the time to play around with them, change things, learn how they work, and understand exactly what they can do.

I’ve tried to give you a decent insight into what you need for Linux shell scripting today. The subject goes far deeper than this, though, so I’ve kept things basic and simple to help you learn. While it isn’t too technical, it is a good starting point.

Chapter Nine

Linux Shell Scripting for Functions

When you write your scripts, there is every chance that you will use the same code in several locations. If it is a small code, rewriting it isn't such a big deal but, where you have large code that needs to be used in several places, it gets tedious. The bash shell provides the solution – encapsulation. In this way, the code is encapsulated in a function, and this can be used wherever it is needed throughout the code.

The Basics

When you start to write more complex scripts, there is every chance that parts of your code performing specific tasks will need to be reused. That could be something as simple as displaying a message multiple times to get answers from users. Or it could be more complex, perhaps complicated calculations that need to be used repeatedly in the script. No matter what it is, having to write the code over and again starts to get tiring, but luckily, the bash shell makes it easy. We simply write the code once and wrap it in a function. Name it according to the contents and, when you need that piece of code anywhere in the script, you can simply call the function name.

Function Creation

Functions can be created in two formats. The first is using the keyword function with the function name assigned to it. The function's name attribute defines the function, but the name should not be one of the reserved keywords; it should be unique and relate to the function contents. The commands are made up of several bash shell commands and when the function is called, the commands are executed in the order they were written in the function.

The second format follows the same function creation pattern used in other programming languages.

```
$ cat test15567
#!/bin/bash
function Myfunction1 {
echo "You are looking at the example of a Linux shell function."
}
count=1
while [ $count -le 10 ]
do
Myfunction1
count=$[ $count + 1 ]
done
echo "You have reached the end of the loop."
Myfunction1
echo "This is the finishing point of your Linux shell script. Where do you want to go now?"
$bash -f main.sh
```

You are looking at the example of a Linux shell function.

You are looking at the example of a Linux shell function.

Repeated eight more times, until the final line of:

You have reached the end of the loop.

You are looking at the example of a Linux shell function.

This is the finishing point of your Linux shell script. Where do you want to go now?

When the name, Myfunction1, is referred to, the bash shell returns its definition and the commands defined in it are executed. The definition does not need to be first in the shell script but if you try using a function before the definition, an error message is thrown:

```
$ cat test15567
#!/bin/bash
count=1
echo "You are seeing this line that comes before the function definition."

function Myfunction1 {
echo "You are looking at the example of a Linux shell function."
}

while [ $count -le 10 ]
do
Myfunction1
count=$[ $count + 1 ]
done
echo "You have reached the end of the loop."
function Myfunction2
echo "This is a definition of another function"
$bash -f main.sh
```

You are seeing this line that comes before the function definition.

You are looking at the example of a Linux shell function.

Repeated nine more times, until the final line of:

You have reached the end of the loop.

main.sh: line 1: $: command not found

main.sh: line 17: syntax error near unexpected token `echo'

main.sh: line 17: `echo "This is a definition of another function"'

Food for thought

When we defined the first function, the shell was defined following a series of commands. When we used Myfunction1 in the script, it was immediately located by the shell. However, our script attempted to use Myfunction2 before it had been defined, throwing an error message.

The point here is that a function must be defined before it can be used in a script and you need to be careful about function names. These must be unique, and they must not be keywords. Where a function is redefined, the new definition overrides the previous one. What is important here is that we didn’t get an error message on the screen but something has definitely gone wrong; finding out what it is will be hard and these are all things you need to keep in mind when you start writing Linux scripts.

```
$ cat test15567
#!/bin/bash
count=1
function Myfunction1 {
echo "You are looking at the example of a Linux shell function."
}

Myfunction1

function Myfunction1 {
echo "This is a definition of another function"
}
Myfunction1

echo "This Linux script ends here."
```

$bash -f main.sh

You are looking at the example of a Linux shell function.

This is a definition of another function

This Linux script ends here.

Returning a Value in Linux Functions

The bash shell tends to treat functions such as mini-scripts that have an exit status. You can generate exit status for functions in three different ways. The exit status by default is something that is returned by the final command in a function. After the execution of the function, you may use the standard $? variable to determine its exit status.

Passing Parameters to a Function

The bash shell treats functions like mini-scripts, which means that you may pass parameters to a function like a regular script. Functions may use a standard parameter environment variables to represent the parameters that are passed to a function on the command line.

```
$ cat test6
#!/bin/bash
# In this example, I will be passing parameters to a function
function addemdigits {
if [ $# -eq 0 ] || [ $# -gt 2 ]
then
echo -1
elif [ $# -eq 1 ]
then
echo $[ $1 + $1 ]
else
echo $[ $1 + $2 ]
fi
}
echo -n "I am now Adding 30 and 85: "
valuedigits=`addemdigits 30 85`
echo $valuedigits
echo -n "Shall we try adding a single number: "
valuedigits=`addemdigits 10`
echo $valuedigits
echo -n "I am not trying to add any numbers now: "
valuedigits=`addemdigits`
echo $valuedigits
echo -n "Finally, I am trying to add three numbers: "
valuedigits=`addemdigits 40 15 90`
echo $valuedigits
$bash -f main.sh
```

I am now Adding 30 and 85: 115

Shall we try adding a single number: 20

I am not trying to add any numbers now: -1

Finally, I am trying to add three numbers: -1

The addemdigits function checks the number of parameters that are passed to the function. If there are no parameters or more than two parameters, the value is returned as -1. If there is a single parameter, it adds that to itself for the result. If there are two parameters, it adds all of them for the result. Since the function uses a special parameter environment variable for its parameter values, it cannot access the script parameter values from the command line of the script.

Passing Arrays to Functions

Passing arrays into functions is a precise art but, to start with, it may be confusing. If an array variable is passed as a single variable, it will not work:

```
$ cat badtest3
#!/bin/bash
function func555 {
echo "The parameters being used in the function are: $@"
myarray=$1
echo "The array that is received is as follows: ${myarray[*]}"
}
thisarray=(1 2 3 4 5 6 7 8 9 10)
echo "This is the original array : ${thisarray[*]}"
func555 $thisarray
$bash -f main.sh
```

This is the original array : 1 2 3 4 5 6 7 8 9 10

The parameters being used in the function are: 1

The array that is received is as follows: 1

If you try to use the array variable as a function parameter, the function takes the first value in the array variable. Fixing this requires that the array variable is broken down into its individual values, which are then used as function parameters. You can reassemble those parameters in the function as a new array variable. Here’s an example:

```
$ cat test10
#!/bin/bash
function testingfunction {
```

```
local myarray
myarray=(`echo "$@"`)
echo "This the new value of the array: ${myarray[*]}"
}
thisarray=(1 2 3 4 5 6 7 8 9 10)
echo "This the original value of the array ${thisarray[*]}"
testingfunction ${thisarray[*]}
```
$bash -f main.sh

This the original value of the array 1 2 3 4 5 6 7 8 9 10

This the new value of the array: 1 2 3 4 5 6 7 8 9 10

The variable named $thisarray contains the individual array values, placing them on the function's command line. The function then builds the array variable using command line parameters and, once you are in the function, you can use the array as you would any other array.

```
$ cat test11
#!/bin/bash
function addingthearray {
local sum=0
local thisarray
thisarray=(`echo "$@"`)
for values in ${thisarray[*]}
do
sum=$[ $sum + $values ]
done
echo $sum
}
iarray=(1 2 3 4 5 6 7 8 9 10 11 12 13)
echo "This is the original value of the array: ${iarray[*]}"
arg1=`echo ${iarray[*]}`
theresult=`addingthearray $arg1`
echo "This is the final result : $theresult"
```
$bash -f main.sh

This is the original value of the array: 1 2 3 4 5 6 7 8 9 10 11 12 13

This is the final result: 91

Handling User Input

You have learned how to write shell scripts that interact with data, files, and variables on a Linux operating system. However, sometimes you need to write a script that must interact with a person who is running the script. The shell gives us several methods to retrieve data from people, including some command line parameters, command-line options, and reading input from the keyboard. The chapter will walk you through the process of incorporating

different methods in the bash shell scripts to get data from the person who is running the script.

Variables called positional parameters are assigned to the command line parameters by the bash shell, including the name of the program executed by the shell. We use standard numbers for the positional parameter variables - $0 is the program name, $1 is the first parameter, $2 the second one, and so on up to $9.

Conclusion

Now that you have reached the end of the book, you should have a good grasp of Linux command line and shell scripting. Unlike other operating systems, the Linux operating system is complicated. You don’t get to see a graphical interface where you can work with a mouse. It is the “all-keyboard” thing that makes Linux hard to learn. However, it is not that hard with this book in your pocket. You can use this book as a reference guide whenever you operate the Linux command line or jump to the editor to write shell scripts. Being consistent and regularly practicing with codes will help you learn the commands and codes faster than just reading the scripts.

Keep this book by your side and you’ll always have keyboard shortcuts and commands at your fingertips. Now, let’s get started with Linux!

References

22 best Linux text editors for coding {2020 Reviews}. (2020, July 12). Knowledge Base by phoenixNAP. https://phoenixnap.com/kb/best-linux-text-editors-for-coding

Bash while loop examples. (2020, November 5). nixCraft. https://www.cyberciti.biz/faq/bash-while-loop/

Blum, R. (n.d.). *Linux Command Line and Shell Scripting*. https://inf.ocs.ku.ac.th/Download/Wiley.Linux.Command.Line.and.Sh

Broida, R. (2017, February 9). *How to install Linux*. CNET. https://www.cnet.com/how-to/how-to-install-linux/

Ward, B. (n.d.). *How Linux Works*. index-of.es/. https://index-of.es/Varios-2/How%20Linux%20Works%20What%20Every%20Superuser%20Sh

LINUX
LINUX SECURITY AND ADMINISTRATION

Andy Vickler

Introduction

If you are new to using Linux, it will be difficult for you to find the right information online. This book has all the information you need to help you install the operating system and show you how you can use it either on your system or a virtual system. The book also has information about how you need to configure user access and other information to maintain the network and server's security on the Linux system. You do not have to know anything about Linux before you use it since the information in this book will guide you every step of the way!

The book introduces the idea of using Linux on a virtual system and provides information on the different distributions of Linux. You can use this information to determine which distros work best for you and download that onto your system. You will also learn about the importance of a root account and the other accounts on the server. The book also provides information about the methods used to control access to users. You will learn how you can grant and revoke privileges to users to help you protect the data.

The book covers how you can secure the information, files, and folders in the operating system. You will be introduced to a list of tools you can use to secure the data on your systems and how you can encrypt and decrypt information using these tools. You can also use passwords to encrypt and decrypt the files and folders on the server and network if you need to.

Since Linux is an operating system, and the data is stored on a server or network, you need to test the network and server's strength. This book will shed light on the method you can use to identify any vulnerability in the system. It will also let you know how you can use scanning to identify the holes in your system. You can use the information in this book to determine how to overcome those vulnerabilities.

Thank you for purchasing the book. I hope you learn more about Linux and how you can protect the information in your files and folders.

Chapter One

Using Linux on Virtual Machines

Have you wanted to use Linux but did not want to use it on your system? Your system may have trouble if you use dual-booting, and the best thing to do is to use Linux on a virtual machine. It is easy for you to use Linux on a virtual machine if you use Windows. The procedure is straightforward. In this chapter, we will look at installing and using Linux on a virtual machine, a VMware Workstation specifically.

If you want to use a virtual machine, you need to find a PC, which allows you to use virtualization. You may have tried to install Linux on your system using a CD, but you may not be sure about dual booting. You should install the Linux operating system on your PC but using a virtual machine.

Virtual machines are environments that replicate the conditions of the hardware on your device. The environment mirrors everything in your personal computer and is limited only by the system's different components. This means you cannot expect to have a four-core CPU on a processor which only has two cores. You can achieve virtualization on multiple systems, and the result of this will be superior on computers that have CPUs that support visualization.

You can use different virtual machines to install the Linux operating system on your computer. VMware is one of the leading manufacturers of virtual machines and applications. In this chapter, we will look at how you can install Linux OS in Windows using a workstation player designed by VMware.

Installing a Workstation Player

If you want the workstation player, you need to download the latest version from the VMware website. They constantly upgrade their workstation player application and tool. For this example, we will use the VMware workstation

15 player, and this file is 150 MB in size. The latest versions can be heavier, so make sure you have good Internet connectivity.

These workstation players are available for home, non-commercial, and personal use and are free. Non-profit organizations and students can use this version since they do not have to shell out any money on installing the operating system. A VMware workstation player performs all the functions a standard virtual machine must. You can also use VMware products since each product offers a wide range of visualization solutions that you can use for any business. If you want to learn more about their products, you can read about them on their website. After you download the VMware workstation player, click on the installer and follow the installation wizard steps to set up your virtual workstation. It is recommended that you download the Enhanced Keyboard Driver during the installation since you may not need it now but will need it later. Complete the installation and reboot your system when the wizard prompts you.

Choose the Correct Distro

You should read about the different Linux distributions available to you and choose the one that works best for you. Some Linux distributions work best on virtual machines, while others cannot work on them. All 64-bit and 32-bit versions of Linux work well on virtual machines. You cannot run Linux distros, such as Raspberry Pi and other ARM architecture Linux distributions on virtual machines. If you want to use an ARM Linux environment on your Windows machine, you should try QEMU. If you do not know which distro you need to choose, choose any from the list below:

Linux Distros

Since there are many options available, you may find it hard to choose the right Linux distro for your system. How do you know that is the best one you should use for your system? What if you want to game using Linux distros? Do you want to use a pretty distro that uses the same structure as macOS?

In this section, we will look at the different Linux distros lists available for you. These distros have been used actively by various individuals over the last few years. It is best to download a Linux distro that you can use safely on your system. You should also check if the distro you use is updated regularly using security patches.

Business Linux Distros

Red Hat Enterprise Linux

This distro is like Fedora, but it is used commercially. This distro was designed for enterprise customers. You can use any of the different addons and variants. If you want to be an administrator, you need to be certified.

SUSE Linux Enterprise

This Linux distro version is designed for an enterprise and can be used by businesses. It is for this reason this variant is easy for one to use with different office programs. You can run this distro on various devices, and it can be used even on critical systems. Many versions of this distro are also available on the Linux website.

Gaming Linux Distros

SparkyLinux Game Over Edition

SparkyLinux has various versions, but this version which focuses on gaming, is the most used. This gaming version comes with various pre-installed games, an LXDE desktop, PlayOnLinux, Steam, and Wine. There are numerous premium and free games available on this distro which you can use easily.

SteamOS

Many gamers have started using Linux as their operating system since it comes with a Steam client. It is easier to install the SteamOS version of Linux if you are a gamer. One of the best Linux distros you can use for gaming is SteamOS, and this is optimized to perform well in any game and comes with in-built sound drivers, proprietary graphics, and a Steam client.

General-Purpose Linux Distros

Ubuntu

This is a Debian-based operating system, and it uses GNOME as the desktop environment. You cannot update this environment since it is used as the default. This Linux distro has regular patch updates, and it improves with

every new release. The latest versions of this OS are designed for hybrids, desktops, and laptops. Therefore, if you are moving from Windows to macOS, you need to use the Ubuntu OS.

openSUSE

This distro is a general OS built by Linux for various projects, but it is primarily used for openSUSE projects. This distro is used both by beginners and by experienced Linux users. This distro comes with an administration program called YaST which controls and monitors the installation, package management, and other functions.

Fedora

This Linux distro was developed by IBM-owned Red Hat and uses a default GNOME desktop environment. You can switch to LXDE, XForms Common Environment (Xfce), Cinnamon, KDE and MATE, and other desktop environments. Some variations of Fedora, like Fedora spins, can be used by people who have specific requirements.

Debian

Debian is an old Linux distro and is the best version compared to other Linux distros. This also comes with a default GNOME desktop environment, but it can also be used in the FreeBSD kernel. Developers are working on making this compatible with other kernels like the Hurd. Some Linux distros, such as Raspbian and Ubuntu, are based on Debian.

Slackware Linux

This is another distro that has been built specifically for simplicity and security. It is a distro that is Unix-like and is used for server and file management since it has web, FTP, and email servers available for use. If you have never tried managing a server or using a Unix server, you can use this server as a live disc. You can also use this as a virtual machine to learn how to use Linux distros better.

Mageia

This Linux distro was developed by a non-profit fork community and had various features that a major desktop environment should have. The default

desktop used by this distro is GNOME and KDE.

SparkyLinux

This version of Linux evolved when the developers were testing the Debian version of Linux. This edition of Linux has a customized lightweight LXDE desktop. You can also use this with other customized desktops.

Gentoo Linux

You can use this distros version on any desktop. It is compatible with multiple requirements, and its performance and versatility make it one of the best versions of Linux OS. Gentoo Linux has Portage, that is an advanced package management system. Since Gentoo can be used on different systems, you gain complete access to your system and control it the way you need to. It does, however, become a problem for a newcomer.

CentOS

Community Enterprise Operating System or CentOS is a distro built by the Red Hat community and is a rebuild of the Red Hat Linux Enterprise. This is a free version of the distro. If you do not want to work with different Linux distros, you can use the Red Hat enterprise at work and the CentOS at home.

Lightweight Linux Distros

Linux lite

The Linux lite distros are based on the Ubuntu LTS releases and have a very minimal footprint. It uses a simple and clean Xfce desktop. This distro also uses a simple Windows-style Start menu which makes any Windows user feel at home. This distro has a small resource footprint, which means you can use it on a PC with 512 MB RAM and 700 MHz CPU. It is for this reason this version is called light. You can use this version on an old computer or on your laptop if you want to maximize battery life.

Lubuntu

This is another lightweight version of Linux based on Ubuntu, and it is perfect to use on laptops or desktops. It uses a lightweight desktop environment and has in-built lightweight applications which are designed for

speed and energy-efficiency. You can use this OS on old mobile devices, computers, and netbooks. It does not need high-speed RAM and has few system requirements. If you want to purchase the best operating system to maintain your device's battery life, you should definitely pick this.

Xubuntu

This derivative of the Ubuntu distros uses an Xfce desktop which means it is lightweight and elegant. You can use it on different notebooks and laptops. If you have devices with low specs, you can use this distro on them. Since it is light, it does not need many system resources. It is for this reason you can use it on old devices, as well.

Puppy Linux

Puppy Linux uses a small distribution that you can run using RAM. This means this version is great for older laptops or computers, and you can use this distro even on computers or laptops without hard drives. Most companies and individuals use this tool to remove malware.

Manjaro Linux

This Linux distro is an easy-to-use, fast and lightweight distribution which mirrors the Arch Linux distro. This version uses the benefits of Arch Linux and is more accessible and user-friendly. It is for this reason a beginner can also work with Manjaro Linux. The default desktop used is Xfce, but you can switch to other options depending on what you are comfortable using.

Arch Linux

This version of Linux is a distribution developed with user experience in mind. The Arch Linux distribution is aimed at keeping things simple and easy. It is updated regularly through patches. Arch has Pacman, a custom-made package manager used in Linux, making it easy for users to build, share and modify packages. This distro is recommended if you are a beginner, since it requires some hands-on experience with the operating system.

NuTyX

Do you want to customize the system you currently use? If yes, you should use this distro. NuTyX allows you to ship bloatware-free and barebones

across the OS. You can also customize the OS using the concept of collection. You have a choice for everything you want to use. For example, you may find a selection of window managers or desktop environments, and you can choose to use the one that works best for you. Using these choices, you can develop a user-determined operating system that has multiple possibilities. You can use this as a focused home theater or a versatile desktop.

Bodhi

This is an Ubuntu-based distribution operating system, and it comes with a beautiful and lightweight Enlightenment desktop. Bodhi can be customized, and it comes with applications and themes. You can use these to expand on the basic operating system you may have downloaded.

Multimedia Linux Distros

Fedora Design Suite

You can save time on installing artistic applications and tools by using Fedora, a spin-off of the Fedora design suite. This design suite comes with GIMP and Inkscape. It also comes with different applications and tools which can be used for art and illustration. This is a distro focused on DTP.

Ubuntu Studio

The Ubuntu studio was released in 2007 and is the default choice used by Linux users. If you are creative, you can use this distro to work on your talents. This distro also comes with an Xfce desktop environment and has low kernel latency. Therefore, everything about this distro is geared towards the production of media. You can use different distros for this as well, but Ubuntu studio is the best for photographers, music producers, designers, and other users.

Linux Distros for Beginners

Endless OS

If you have just started using Linux, you may want to keep everything simple. The best Linux distros to use for this is the endless OS. Families often use this since it comes with multiple applications. It is best to use this

OS if you do not have an Internet connection at home. You can also use this if you are unsure about which application you need to use on your Linux OS. This is not an ideal approach for you if you are an experienced user. If you are new to using open-source operating systems, you can use this OS to obtain more information about working with Linux.

Linux Mint

Linux Mint is a modern and elegant distro that is powerful and easy to use. This distro is based on Ubuntu and is reliable. It was developed with the idea of a software manager in mind. This distro is one of the top-rated Linux operating systems since 2011. Many macOS and Windows refugees choose to use this as their new virtual desktop. Linux Mint also comes with various desktop options. You can use the Cinnamon desktop, that is the default for Linux Mint. Alternatively, you can use KDE, Xfce, or MATE. You can also use a Debian and Linux Mint combination if you are a beginner.

Deepin

Deepin is another Ubuntu-based distro, and it comes with a stylish DDE or Deepin Desktop Environment. This distro is great for new Linux users. It is simple and intuitive and features a variety of system settings panel displays. Deepin is inspired and developed based on macOS. This distro also has a software center that is easy to use. It has tools that are far superior when compared to other Linux distros. For this reason, Deepin is a great operating system to use if you are switching from macOS.

Pop!_OS

This is another Ubuntu-based operating system that was manufactured based on Linux hardware manufacturer system 76. It uses a GNOME desktop as the default environment and has a theme that you can change. The colors vary depending on the brand identity of System 76. This distro also comes with its own application and installation browser, making it easier for you to install the required Linux applications. Some applications may not match the theme, but this is an easy-to-use distro for a beginner.

Zorin OS

If you are new to Linux, you can use Zorin OS since it is designed

specifically for beginners. This distro can ease the transition from using other operating systems to Linux. This operating system is based on Ubuntu and has several applications, which are like Windows applications. This makes it easy for you to use Zorin OS since you know how to work with the applications. You can also configure the desktop on the Zorin OS distro to resemble Linux, Windows, or macOS.

Elementary OS

Another Ubuntu-based distro is the Elementary OS which has differentiated it from other distros greatly since 2013. One of the most common features of this distro is the use of simple and beautiful default applications and tools. These applications also maintain the operating system's aesthetic appeal, such as using the Epiphany web browser and Mail for email.

Elementary OS also has different features you can use to improve the function of various operating system functions. You can use different productivity apps, as well. If you want to change the desktop layout so it matches that of a macOS, you can use the Elementary OS.

RoboLinux

It is difficult for you to switch from Windows to Linux and vice versa because not all applications and tools used are compatible. Various distros in Linux have found a workaround for this issue. The RoboLinux distro, unlike other distros, has a better or easy solution. It allows you to set up a Windows virtual machine easily on your device. You can set up Windows XP and later versions easily on RoboLinux. This prevents dual booting. You can access all your Windows applications anytime you need to.

Kubuntu

There are different variations or derivatives developed on the Ubuntu operating system. Another popular Ubuntu option is Kubuntu, and this distro uses a KDE desktop as the default environment. If you look at the system beneath the environment, it is the same as Ubuntu and has the same releases as Ubuntu.

Raspberry Pi Linux Distros

Raspberry Pi is an extremely common and popular Linux machine, but the

other distros mentioned in this list will not work since Pi uses an ARM processor instead of AMD or Intel 32-bit or 64-bit CPUs. It is for this reason the Raspberry Pi Foundation worked on developing specialist distros. Some of them are Pi-friendly versions of existing Linux operating systems, and these are covered in the sections below.

Raspbian Stretch

For Raspbian Pi, the default operating system used is the Debian-based Raspbian Stretch. The Raspberry Pi Foundation developed the latter. Raspbian Stretch is an ARM Linux distro and has multiple programming tools and applications. A beginner can use these tools to learn more about coding on Linux. Raspbian also has LXDE-based PIXEL environments, making this distro the best option, especially if you are using Raspberry Pi.

Kano OS

Kano OS is like Raspbian, but it focuses more on coding. The operating system is aimed at helping children learn more about how to code. The system comes with an interactive user interface, and this gives your children the tools he needs to code without too much fuss.

DietPi

Do you work on projects where you need to use barebones operating systems? If yes, you should choose DietPi. This is a very light Debian-based operating system and can be used on all models of Raspberry Pi. You can also use it on single-board computers if you have one. Raspbian Stretch Lite is an option that most Pi users choose, especially if they use applications with a small footprint from the selected operating system. The difference between DietPi and other Raspbian Pi operating systems is the amount of space you need to run the OS on your system. DietPi only needs 1 GB storage, while others need 2 GB or more.

Linux Distros for Security and Recovery

Qubes 3.2

I am sure you know Linux is a very secure operating system, and it is better than Windows, too. The most secure Linux distros are Qubes. The current

version is 3.2, and this is a better upgrade when compared to the earlier versions of Qubes. Edward Snowden stated that Qubes is a reasonably secure operating system, and this testimonial is enough for you to use this distro in your virtual machine. If you are a security-conscious user, you need to choose Qubes. This distro is known for freedom, security, and various privacy features. It also comes with sandboxes that allow you to separate the application and hardware when you perform different functions.

Kali Linux

Kali Linux was earlier known as BackTrack. It is a penetration testing Linux distro that is used often by the online security community. This is a Debian-based distro which makes it easier for you to perform various forensic tasks.

Parted Magic

This Linux distro is used to manage disk space on your system. You can partition the hard disk and copy the information across different servers as the OS's primary applications. Parted Magic makes it easier for you to secure erasing and recovery of data.

GParted

GParted is a single-purpose distribution operating system that makes it easier for you to partition hard drives. You can do this easily using a graphical interface. A Linux user is familiar with using standard versions of various distribution operating systems. This is a dedicated and standalone operating system, but it can also be run using a CD. If you want to manage the disk space on your system without booting your computer or the operating system, you can use GParted.

Tails

Tails is a Linux distribution operating system that revolves wholly around the idea of security and privacy. This is an operating system that you can use through a USB stick, SD card, or DVD. You can use this operating system anywhere without worrying about leaving a trace. You route everything you perform on the system through a different router (called the Onion Router or TOR) to maintain anonymity. You can also use different cryptographic tools to protect all the information you send or receive from prying eyes.

Bruce Schneier, a famous American cryptographer, loves using Tails, and this is a big endorsement.

You are looking for a new way to use a secure and portable tool. But how do you know that is the best Linux distro for you to use? There are so many operating systems you can choose from, but you need to choose the one that does what you want it to. Since there are Linux distros for different purposes, you can choose which works best for you. If you want to carry a distro with you on a USB stick, you need to round up the best portable Linux distros.

Setting Up the Virtual Machine

While the ISO for your Linux distros is downloading, you should spend time configuring your virtual machine settings. To do this, you need to launch the workstation player. Follow the steps given below to create a virtual machine:

Step One: When the installation wizard opens, select the option to create new virtual machines

Step Two: Choose the default option, that is the Installer disc image file (ISO). It is best to do it this way to prevent anything wrong from happening during the installation

Step Three: If you cannot find the ISO file, look for it using the browse option

Step Four: Select ‘guest OS’ and move to the next step of your installation

Step Five: You need to select Linux as the type of guest operating system

Step Six: Select the version of the OS from the list

Step Seven: Click on the next option to move to the next step, and key in the name of the virtual machine

Step Eight: Confirm the location where the OS files need to be saved and let the installation wizard complete the process

Once the operating system is selected, and the installation wizard is configured, you need to build your virtual machine. Follow the steps given below to do this:

Step One: Choose the maximum disk size from the option to select the specific disk capacity. You can stick to the default if you want to

Step Two: You can also choose to split the virtual disk into multiple segments. It is recommended to do this since you can move the virtual machine easily onto your new system

Step Three: Once you are happy with the details, you should confirm them and move onto the next step

Step Four: Click finish to install your virtual machine

The Linux virtual machine will now be a part of your VMware Workstation player.

Customizing Virtual Hardware

When you download the virtual machine, you can customize it before you install Linux. You may need to do this for the following reasons:

1. The Linux distros you are using may need a specific type of OS
2. There may be some components missing in your operating system

You can fix this easily by accessing the settings on your virtual machine. This is where you can work with the virtual machine's hardware and tweak the settings in ways that are beyond the HDD. You can make changes to the network adapter configuration, memory, processors, and more.

You need to look at the processor screen before anything else. You will find a reference to the virtualization engine. This will work automatically when you install the virtual machine in your system. If you want to troubleshoot your processor when there are issues, you need to set the AMD-V or Intel VT-x.

The memory screen on your virtual desktop settings will help you address memory performances. Your virtual machine will have the details of the RAM you need to use. It will tell you about the settings you need to maintain on your physical and virtual machines. It is recommended for you to stick to these settings. If you reduce the memory size below or higher than the recommended size, you may either create a problem or slow your system.

This will mean your system cannot function the way it should.

You then need to spend time on the display settings. You can stick to the default settings if you want, but you can play around with the 3D toggle acceleration if there is an issue. You can set up different monitors differently depending on what you would like to use them for. It is important to note that some modes you set up will clash with the different desktops you use.

If you are happy with the changes, confirm them and click on the play button to start your virtual machine.

Download and Install Tools

Now that you have downloaded your virtual machine, you should reboot it. You will be asked to look at different VMware tools you can download and install at this stage. All you need to do is to agree to what the wizard says and let the tools install on your virtual machine. These tools will improve your virtual machine’s performance, enabling you to share folders between the guest and host machines, the virtual and physical machine, respectively.

Installing Linux on VMware

When you launch your virtual machine, the ISO boots directly into your live environment, this version of Linux is only temporary and will not reside in your system memory or boot media when you turn off the device. If you want to use the environment the way it should be used, you should use the virtual machine's install option.

At this point, the installation wizard will continue to install the operating system on the actual machine. You need to follow the installation wizard steps, create an account, and set the other options when you are prompted to do this. When the installation is complete, you can use the Linux virtual machine and use the guest operating system. The process is this simple.

Running Linux on a Virtual Machine

Now that you have the virtual machine launched on your system, you can run the distros you want to use on the virtual machine using the Play button on your system. If you are looking for software to install, you can choose the preinstalled applications which come as part of your Linux distros. You can

also choose to download different apps if you want to.

If you want to access the Linux terminal through your system, you can do this without a VMware workstation.

Installing a Linux Distro on a Windows Virtual Machine

The easiest way to access Linux on your system is through a virtual machine if you use a Windows operating system. The workstation player from VMware has the best tools to help you do this. You can install Linux easily on your VMware workstation. Let us quickly go through the steps again:

Step One: You need to download the VMware workstation player. It is recommended that you download the latest version

Step Two: Install the workstation player and reboot windows

Step Three: Configure and create the virtual machine

Step Four: Install the Linux distros of your choice on your machine

Step Five: Reboot the virtual machine and use the operating system

The process is this simple, and you do not have to stick to one operating system alone. You can choose from the list of Linux distros mentioned above, and you can install all of them on your workstation player.

Chapter Two

Securing User Accounts on Linux

Don’t Login Using a Root Account

The root user on your Linux operating system is like the administrator user on Windows. There are situations where you may want to log in to the administrator account on Windows. You should never log in to Linux using the root account or user. Microsoft has worked on improving the security practices followed on Windows using user access control (UAC). Linux does not have any such tools readily available, so you should avoid using the root user.

Using Sudo Accounts

Users need to avoid using the root account to access the operating system. Ubuntu gives sudo access to different users. Ubuntu locks the root account, so any user cannot log in to the root account without going out of his way to enable the root account and obtain the password.

Earlier versions of Linux distributions allowed users to access the root account through the graphical login screen. They can get to the root desktop. Many applications may not run as root applications, while others may throw errors. If you are moving from using Windows to Linux, you may want to log in to the operating system using root. You may think this is no different from using an administrator account on Windows. This is a terrible thing to do since it endangers your system.

When you use sudo accounts on Ubuntu, you get root privileges. If you use su, you only access the root shell to run the command you may need to use before you come out of the root shell. Through sudo, you can enforce the right security practices in your system. These accesses also ensure that you run the command as a root user without exiting the root shell. You can access and run an application as a root user only when you are in the root shell.

Reducing the Damage

When you access the operating system using your personal account, the system will ensure you that none of the programs you run write on the other programs or systems on your hard drive. The programs you run using your personal account only make changes to your home folder. You cannot modify any system file without gaining access to the root account. This is an easy way to maintain the security of your computer.

Let us assume you use Google Chrome as your default web browsing application. If there is a vulnerability in chrome, and you run the operating system using a root account, you will leave your system open to malware and other web pages. Malware can be pushed into your system through different web pages, and it can be used to read various files on your system, including those stored on personal folders. They can also compromise different system functions. If you are logged into your system using a limited user account, the malicious web page can still send malware, but it cannot read the source files on the operating system or replace any commands. The malware will only damage the information on your account folder. This can lead to problems, but it is better than having your entire system compromised.

Using a limited access account will protect your system against buggy and malicious applications. For instance, you may have an application that has a bug that deletes the files it can access. So, when you run it, the application removes all files from your account folder. This is definitely not good for you, but you can restore the files easily if you have a backup. If the application can access the root account, it can delete every file on your hard drive. This means you need to reinstall the full operating system again on your computer.

Fine-Grained Permissions

Older distributions of Linux used the root account to run various administration programs on your system. Modern distribution systems use tools, like PolicyKit, to differentiate the access or functions that different user accounts on Linux can perform. You can determine the permissions which can be defined in the application, too.

For instance, if you have a software management application on your operating system, you can use PolicyKit to ensure the application can only

install and update your software whenever necessary. The application would only run in your operating system based on permissions you have set. Only the part of the different programs in the operating system which use this software will have elevated or higher permissions. Only this part can install and manage software in your operating system.

The program you are using will not have root access which means it cannot create a security hole or vulnerability in the application or system. You can also use PolicyKit to ensure only some users have the authority to make changes to the system administration. You can also ensure these users do not have access to the root account and cannot make any changes to the operating system. This allows you to control who can perform activities on your system without any hassle.

You can log in to the graphical desktop using your root account in Linux. You can also delete every file on your hard drive or write random information or noise into your hard drive when your system is running. This will obliterate and ruin your file system. This is a terrible idea. You may know what you are doing, but your system is not designed to let you run the applications and tools using the root account. If you use the root account, you will bypass security, leaving your operating system open to any hacks.

Managing User Account Security

You need to maintain control over the user accounts if you want to secure the operating system. This section will look at how you can perform certain user account management functions and tasks. We will also look at how you can implement security measures.

Adding New Users

As mentioned earlier, you should avoid using a root account to perform any functions, especially any regular operations. If you have deployed the operating system on a new server or virtual machine, you will only have a root account on the server. This means you need to create new accounts for everybody who will access the operating system. If it is only you, create a username. To do this, run the command in the shell: *adduser <username>*.

You also need to create a password and enter other key information about the user, and you can do this using the user creation procedure. On Red Hat and

CentOS variants, you need to unlock the account to create a password. Do this using the following command: *passwd <username>*.

If you want to use the account you have created to manage the system and other files, you need to give it sudo privileges. If you are using an Ubuntu-based Linux OS, you can do this using the following command: *adduser <username> sudo*. When you add sudo permissions to various users on a CentOS variant, you need to use a different command: *gpasswd -a <username> wheel*. A Debian user does not have to add any sudo privileges since those may have been given to the account as a default setting. If these privileges are not provided, use the command: *apt-get install sudo*. After you do this, you can use the above commands to add the account to the sudo user list. It is important to note the changes to the privilege user groups will only change when you log in after the setting has been updated.

If a user is given sudo permissions, he can perform the operations a root account is allowed to perform. Since the operations are performed in a root shell, you do not compromise on security. If you need to add more users to the server, give them sudo access. Do not give them root access since it is dangerous to share the password with every user. It is also recommended for you to use a sudo account over a root account.

Disable Root Login

Once you set up your account on Linux, you need to disable remote access for the root account. You must do this if you want to prevent any security breaches. The configuration file has the OpenSSH server settings, and you can open it using a text editor on Ubuntu or Debian using the following command: *sudo nano /etc/ssh/sshd_config*. If you use Red Hat or CentOS variants, you can use the following command: *sudo vi /etc/ssh/sshd_config*. You can also search for different authentication settings and options and change the permission to log in to the root account. You can do this by changing the setting to no using the command: *PermitRootLogin no*.

Save the file with the commands. Once you make the updates, restart the server. If you use cloud servers and CentOS, you can use the command: *sudo systemctl restart sshd*. On systems where you use Ubuntu, call the configuration file and restart the system using the following command: *sudo service ssh restart*. The same command will work on Debian, as well.

Password Policies in Linux

If you allow users to access the server remotely, you need to enforce and implement password policies using a module in Linux called PAM. You can invoke this using *pam_cracklib.so*. You can use this module to check the passwords entered by users against any dictionary words. This prevents the use of weak passwords. The module also allows you to define any new password requirements, such as complexity and length. On Debian and Ubuntu systems, you can install this module using the command: *sudo apt-get install libpam-cracklib*.

Red Hat and CentOS variants come with this module installed. Once you have the distros installed, you need to open and edit the configuration file. If you use Debian or Ubuntu, open the file in a text editor. You can do this using the following command: *sudo nano /etc/pam.d/common-password*. The file will be stored with a new name on CentOS. Use the command *sudo vi /etc/pam.d/system-auth* to access the file.

When you install the module on Debian or Ubuntu, the password is configured on the server, and the checks are performed. Therefore, you need to find a setting corresponding to this and change it, so it looks like the following: *password required pam_cracklib.so retry=3 minlen=8 difok=3 dcredit=1 ucredit=1 lcredit=1*. If you use CentOS, you may need to add the entire line above directly to the configuration file, but this is dependent on the version you are using.

Let us look at the parameters and see what they mean:

1. **Retry:** This parameter defines the number of times the user can enter an incorrect password
2. **Minlen:** This parameter defines the minimum length every password should have
3. **Difok:** This parameter is used to check the number of characters reused in the current password when compared to a previous password
4. **Dcredit:** This indicates the numerals you need to have in your password

5. **Ucredit:** This defines the number of uppercase characters
6. **Lcredit:** This defines the number of lowercase characters

You can update these settings and save the updated settings. You need to understand that the policies will apply to every user on the server except for administrator users. The latter is responsible for maintaining the strength of root passwords.

Restrict SSH Access

You can use the OpenSSH server to limit the number of connections to the server based on the group users is categorized into. This is a useful technique to have up your sleeve, especially if you have many users and you have some who should not access your server remotely. You can also do this to add more security to your server. For example, when you run a web service, you may not want everybody to access that service using the same server. To do this, you need to create new user groups. Use an apt name for the group. In the following command, we will create a group of users called sshusers: *sudo groupadd sshusers*.

If you want to be a part of this group, add your username against the group's name using the command: *sudo gpasswd -a <username> sshusers*. If you want to check who has been added to the group, you can use the command: *groups <username>*. The output of this command will show you the different groups of which the username is a part. You can also include another user group in the list for the same user. The list will also have a username that is the same as the name of the user.

The output of the above command is: *user: user sudo sshusers.*

Once this is done, you can specify the group you want to use for OpenSSH. To do this, you need to open the configuration file using an editor. If you have nano installed in your system, you can use the command: *sudo nano /etc/ssh/sshd_config*. Otherwise, you can use the command *sudo vi /etc/ssh/sshd_config*. At the end of your configuration file, you need to add the following line: *AllowGroups sshusers*.

Ensure the configuration option you have chosen is not commented out using the hashtag sign in front of it. Save the file down and exit the editor. Now,

restart the SSH server and use this command if you use Debian or Ubuntu servers: *sudo service ssh restart*. With Red Hat and CentOS variants, you can do the same using this command: *sudo systemctl restart sshd*.

Using the new configuration, a user who does not belong to a specific group can be denied access to the server over SSH. Their passwords may be entered correctly, but they will not be given access. This reduces the chance of people hacking the server through brute force attacks. This method is the easiest way to protect the Linux server you are using.

Understanding Account Privileges

As a Linux security professional or system administrator, you need to manage the system users and their activities. You also need to limit the activities a user can perform on the system using different tools. It is important to remember that a user can make mistakes. These can be anything, including assigning an administrator's privileges to a user who is new to Linux, running an application or script on the system using root privileges, or leaving vulnerabilities in your system that allow a hacker to access it. Since you are responsible, you need to do everything in your power to ensure users do not perform any activity on the system that they should not. You may not even want to clean up the mess.

Your main goal should be to control what a user can and cannot do when he accesses the Linux server. A user on a Linux system needs an account to access the system. Users can access these accounts directly when they perform any activity on the operating system, data, or application. Every system requires specific accounts so that it can function properly. It also needs applications that will need an account for them to access the server.

You can manage the privileges assigned to users in multiple ways and using different tools. The best way to manage users' access and privileges is using the privileged access management tools since these allow you to manage the activity centrally.

In this section, we will look at nine important practices you need to bear in mind when it comes to managing accounts and privileges on Linux. It is important to bear in mind that these are not the only ways to protect accounts. Newer methods are emerging regularly, and you can learn more about them

depending on the type of system you maintain.

Manage Linux User Accounts

It is easy to create an account on Linux, and we have seen the commands you can use to do this in the previous section. It becomes difficult to remove or disable an account that you no longer need. If you do not remove these accounts, they continue to be active, which can cause further issues. Every Linux system needs an account that users can use to access the server to perform any activity on the operating system or on the data stored. The system will need accounts for it to function properly. Applications also require accounts that can be used to connect it to the server.

Most people assume that account management only revolves around managing user accounts. They do not worry about the functional, system, or application accounts. This leaves a lot of gaps in your risk management for the server.

Reducing Privileges

When it comes to privileged accounts, you need to restrict these users' access and rights to ensure they only have the authority to perform the required activities and nothing more. It is important to ensure users only do what they are required to do. If you give a user more access than he needs, it will be difficult for you to control what a user can do. You need to consider the following when you assign privileges:

1. Do system administrators need to use the root account?
2. Does a system administrator need to run commands which only the root account can run?
3. Does a database administrator need to access the root account?

Managing Passwords

You may have implemented and enforced a password policy and the parameters one needs to adhere to when they set their policy. Having said that, the goal should be to reduce the number of passwords used in the system. There are places and times where you need to use passwords still. If you use passwords on the system, you must find a way to manage them.

The following need to be kept in mind when it comes to managing passwords on the system:

1. Ensure passwords are frequently changed
2. Prevent the use of old passwords
3. Assess the complexity of the passwords used in the system

These are the basics when it comes to managing passwords. You need to expand this to all passwords on the server, including the root, user, functional, service accounts, and application passwords.

Reduce the Use of Shared Accounts

As an administrator, you should never share an account with another user on the server. Administrators need to have separate accounts for accountability and traceability for all the actions performed by them. You need to track every user's actions. Every Linux system comes with a root account. Applications on your server will have accounts associated with them, and these accounts have their own set of privileges. This limits the access the application can have to the system.

There are times when a data administrator or application administrator needs to perform privileged activities on the operating system, data service, or application. You need to implement and enforce a system where a user can access a different account without logging into the privileged account. This is a sudo command which can be used to access the operating system. You, as an administrator, can extend a user's privilege without allowing them to share accounts. It provides the required accountability and traceability to identify the activities they perform.

Control Access to Accounts

It is important to understand that not every account present in the system needs to be accessed over a network. You must implement and enforce privilege control across the server. It is only when you do this that nobody logs into incorrect accounts or performs unauthorized activities on your server. It is also important to ensure that every user does not access the server using the SSH configuration. Not every user account on the server needs to access the server from different computers on the network. You do not have

to give a user access to the server from anywhere. Users should only access the server to perform the tasks he is assigned.

Maintain Logs

It is important to log everything that happens on your server. You need to log the accesses, changes, patches, privileges, and features of every user on your server. This is not an exhaustive list, and you can look at other aspects of a user's activity, as well. It will take you and your team time to go through the messages and evaluate all the activities performed. This is something you most certainly need to do.

You should report which user has access to perform which activity. You need this information for different regulations. It is important to log all changes to different accounts, changes to entitlements, deletions or additions of users, or any other activity performed by the users. You can only make changes to your servers' rules if you are aware of the activities being performed by the users.

Record and Manage Privileged Activity

It is important to know which user accessed the server, which host was used, where he accessed the server, and how he did it. It is even more important to know the user's privileges and analyze what the user did to the server. You need to collect this data and analyze it to see what activities are being performed on your system.

If you are the administrator, you can manage user rights, activities, and their activities' timings. This is a powerful tool. To understand what a user is doing on the system, you can run commands using privileges not assigned to the users. Some examples of such privileges are database administrators or application administrators. These users can perform only configuration or maintenance activities. If these users have excessive privileges, it can lead to various problems. You need to manage the administrator access, even if it looks like you cannot trust the administrator. These accounts should be looked at carefully, and the activities should be monitored. This is not because of the people who can use the accounts but because the accounts are a hacker's valuable asset.

Notify or Alert in Case of Suspicious Activity

Once you implement and establish the audit and logging system on Linux, you need to set up an automated notification to alert you of any suspicious activity on any server used in the system. This does not mean you set a signal to report any minor violation, but those which can cause a breach in your servers. Your notification and alerting system must be defined on rules to stop inappropriate behavior. When you log this activity, it will show you that a user is performing a function he should not. You can then use the notification to define rules which will prevent this from happening again. This process will make it easier for you to be proactive instead of reactive. This process is based on the concept of detecting through enforcing. This concept means that you use the same rules to detect and prevent any bad behavior from occurring.

Unify and Centralize

All Linux distros allow you to perform all the actions listed so far natively. For instance, you can use SSH to configure how a user can access the server and where he can access it. You can also use it to define what activities the user can perform using that access. It is easy for you to do this manually if you are working only with one system and a few users. If you have multiple servers, it increases the effort you need to put in to maintain the configuration of the servers. You may also make an error when it comes to manually updating the access and user groups for every user.

It is important to couple privilege management and access control. You will make mistakes if you look at these two domains separately. In the world of Linux security, you need to couple access and account control as well. To do this, you need to have tools that allow you to manage the access control, account management, and privilege management domains centrally.

Chapter Three

Securing Servers Using Firewalls

As a system administrator, you need to learn how to configure and maintain a firewall for your Linux servers. In this chapter, we will look at how you can manage the server using a firewall-cmd command.

If you have worked on network security before, you know that a firewall is an important part of it. Therefore, it is important for you, as the system administrator, to learn about how a firewall works. It is only when you understand how a firewall works that you can secure the server and make the right choice about which traffic you prefer to allow to enter and exit your server. Firewall is a different name, and some people confuse it with a Tron-style battle which happens at the ends or outside of the network where the game is played. Unlike the Tron-style battle where rogue data packets are released to protect a user's data or techno fortress, a firewall is only a small part of software you can use to control the incoming and incoming traffic in the network.

Ports

You can use firewalls to manage any traffic flowing into and out of the server, and it does this by monitoring the ports used in the network. When it comes to firewalls, the word port does not refer to any physical connection port, such as a VGA, HDMI, or USB. In terms of a firewall, ports are artificial connections or constructs which are created in an operating system. This port is used as a way to connect or create a pathway between various data sources and the operating system to move specific data types. You can name the port using different words, but port is the word used too often. What I am trying to say is that there is nothing new or different about a port. It is only a way to move the data to the right place.

There are different ports used in an operating system, and some are used more often than others. These ports are conventions. If you have worked with

HTTP and on coding, you know that the traffic moving via HTTP moves on port 80. Similarly, HTTPS moves on port 443; SSH uses port 22, and FTP uses port 21. When data is moved from one system to another, a prefix is added to the data set or packets moving through the port. This prefix will determine the port that should be used to move the packets. If the end port accepts the data and information you send through the protocol, it means the data is accepted by the endpoint successfully.

If you do not know how this works, you can learn about it by accessing any URL. Open any browser on your system and use the command *example.com: 80* to send a request to port 80 on your system using an HTTP request through your network using the website. When you do this, you will find yourself on a landing page that is a response. You do not have to enter a port when you try to access a website. All you need to do is enter the URL, and you will reach your website since the network will look at the protocol being called and move the traffic to the respective port. If you want to test how this works, you can run the code using a web browser that is based on a terminal.

```
$ curl --connect-timeout 3 "http://example.com:80" | head -n4
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
```

In the same form, use a different port to reject access to a website. For example, you can add port number 79 (using the notation *example.com:79*) to the above notation. This will push the website to access the incorrect port, which will decline access to the website.

```
$ curl --connect-timeout 3 "http://example.com:79"
curl: (7) Failed to connect: Network is unreachable
```

There is a correlation between protocols and ports on a network, and experts set these conventions. You can easily change these settings if you need to on your computer. When people began to work with computers and networks, they were afraid to change the ports in their network or servers because they believed it could lead to an attack. Now, attacks are sophisticated, and this means the hacker will not learn anything by running a port scanner on the network or web server to determine whether or not there is a change in the ports. A firewall will look at the activity that is performed in the web server on any port.

Using the Firewall-cmd Interface

The operating system you are using will have an infrastructure, in the form of a server and network, in the rack. The objective of this interface is to run the firewall. Alternatively, your modem and router may come installed with a firewall installed in your modem or router. This will act as the primary gateway and path used by any website on the Internet. It is important to remember that firewalls come pre-configured and use a different interface that you can change. In this section, we will look at the different commands you can use through the firewall-cmd interface on your Linux distribution.

You can manage the firewalld daemon on Linux using the firewall-cmd interface. This interface connects the Internet to the Netfilter framework on the Linux kernel. This stack will not be found in modems that have a firewall embedded in them. You can use it on any Linux distribution, which allows you to use the command systemd. If you do not have a firewall that is active on your network, you cannot use the firewall-cmd interface to control anything. Therefore, the first thing you need to do is to check if the interface runs on your system:

```
$ sudo systemctl enable --now firewalld
```

Using this command, you can start the firewall daemon on your network and ensure that the daemon will load when you reboot the system.

Bock Everything

When it comes to configuring the firewall used, most experts will ask you to block everything on the firewall. Experts also recommend that once you block everything, you can open the network up to some ports. This means you need to know exactly what your server needs and identify how to do it.

If you work in an organization with its own DNS caching service or DNS, you need to unlock the port before handling the communication in the DNS. If you solely rely on the SSH configuration and use that to change the servers and access them remotely, you cannot block it. You need to know which service runs on the operating system and understand if the service is internal to the organization. You also need to determine if there is any interaction between different services in the server.

When it comes to proprietary software, the server may make calls to external

ports, and you may not be aware of these calls. If there are applications that react terribly to any strict firewall on your server, you need to use reverse engineering. You may also need to speak to the application's support line to understand the type of traffic the application is creating. You also need to know why the application behaves the way it does. When it comes to open-source software, this is not a common issue. This is not true for when it comes to complex software stacks. Most media players, for example, will make calls to the Internet. They will need to use the Internet to fetch album art or tracklists. An example of this type of application is Spotify. You need to have your Internet connection on if you want to listen to different music.

The firewall-cmd has presets, and these are zones. These zones have defaults configured in them, which the server can choose from. When you use these zones, you do not have to build the firewall for your server from scratch. A zone will apply to only one section of the network interface, which means if you have two ethernet cables or interfaces on your server, you will have one zone taking care of one ethernet. Therefore, you need to identify the different zones in your system. If you want to look at all the zones in your system, use the following lines of code:

```
$ sudo firewall-cmd --get-zones
block dmz drop external home internal public trusted work
```

If you want to learn more about what is allowed in a specific zone, you can run the following lines of code:

```
$ sudo firewall-cmd --zone work --list-all
work
  target: default
  icmp-block-inversion: no
  interfaces: ens3
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

When you use an existing zone in your network interface as the point to build your firewall, you make life easier for yourself. Alternatively, you can choose to create a firewall from scratch.

Creating a Zone

You need to use the command –new-zone to add a new zone on your web server. Any action you perform using the firewall-cmd command will only persist until you restart the computer. To make anything permanent on your firewall, you need to use the permanent tag to ensure the settings you write are saved. Let us consider an example where you can add a new zone called corp. If you want to make the corp permanent, you need to use the permanent flag and reload the firewall to save the rules. This means your new zone will be active.

```
$ sudo firewall-cmd - -new-zone corp - -permanent
success
$ sudo firewall-cmd - -reload
```

Now that you have a new zone in place, you can assign a network interface to this. Before you do this, you need to use the ssh service to access the zone externally. You can also make this a permanent change to your server using the - -permanent flag. This will ensure the changes are maintained even when you reboot the system.

```
$ sudo firewall-cmd - -zone corp - -add-service ssh - -permanent
```

You can now use the corp zone since it is active. This will not accept traffic from any port except for SSH. We also have not assigned any network interface to this zone. If you want to activate the corp zone and make it the default zone, you need to use the –change-interface option in the code.

```
$ firewall-cmd --change-interface ens3 \
  --zone corp --permanent
```

The interface is under the control of NetworkManager, setting zone to 'corp'.

```
Success
```

When you make the corp zone the default zone used in the server, every command you use in the code will be applied to the corp zone. The firewall-cmd command will access the corp zone unless you specify any other zone in the code. You can decide if the corp will be the default zone depending on whether you want it to be the primary zone. If you want to do this, you need to use the following lines of code:

```
$ sudo firewall-cmd --set-default corp
```

You can also look at the interfaces and the zones assigned to the interface. To

do this, you need to use the following lines:

```
$ sudo firewall-cmd --get-active-zones``
corp
  interfaces: ens3
work
  interfaces: ens4
```

Removing or Adding Services

You have now blocked access to the network from every protocol except for SSH. Therefore, you can open the ports used in your network based on the protocol being used. The easy and quick way for you to permit or manage traffic through the firewall is to include predefined networks or services. If you want to look at the list of services that are available on your network, you can use the following lines of code:

```
$ sudo firewall-cmd --get-services
RH-Satellite-6 amanda-client amqp
amqps apcupsd audit bacula bacula-client
bgp bitcoin bitcoin-rpc bitcoin-testnet ceph
cockpit dhcp dhcpv6 dhcpv6-client distcc dns
[...]
```

Let us consider a situation where you need to run a web server. To do this, you need to install a web server on the network you want to use. You can use apache2 on Debian or Ubuntu or the httpd package on Fedora or RHEL. In the following example, we will use the httpd service:

```
$ sudo dnf install httpd
$ sudo systemctl --enable --now httpd
```

You should now test the web server from your local network and see how it works.

```
$ curl --silent localhost:80 | grep title
<title>Test Page for the Apache HTTP Server on Red Hat Enterprise Linux</title>
```

You should then use the network and connect to the web server in your network from any browser, preferably an external one.

```
$ curl --connect-timeout 3 192.168.122.206
curl: (28) Connection timed out after 3001 milliseconds
```

Unblocking a Service

If you want to permit the traffic from the HTTP protocol through your network or firewall, you need to include the HTTP service in your web

server.

```
$ sudo firewall-cmd --add-service http --permanent
$ sudo firewall-cmd –reload
```

You should test this inclusion using an external browser:

```
$ curl --silent 192.168.122.206 | grep title
<title>Test Page for the Apache HTTP Server on Red Hat Enterprise Linux</title>
```

Now, you know how you can include or add a service to your web server. It is, therefore, easy for you to remove one.

```
$ sudo firewall-cmd --remove-service http --permanent
$ sudo firewall-cmd –reload
```

Removing and Adding Ports

You may have trouble with finding a predefined service on your system or network. The network or the ports may also assume that the defaults used do not match, and they may reject the access. Instead of adding a service to your firewall, you can add a protocol type or port number using the command --*add-port*. For example, if you want to use a non-standard port for the SSH protocol in your custom zone, you can add the details using the commands below:

```
$ sudo firewall-cmd --add-port 1622/tcp --permanent
success
$ sudo firewall-cmd --reload
```

To remove that port, use --remove-port:

```
$ sudo firewall-cmd --remove-port 1622/tcp --permanent
success
$ sudo firewall-cmd –reload
```

Walls of Fire

You can do quite a bit using the firewall_cmd command, which is not listed above. You can define ICMP blocking, define the sources which can send data and traffic to your system using the network, or even define the services you want to permit. It is easy for you to learn by experimenting. So, you can install Fedora or Red Hat Enterprise Linux in a GNOME box. This will allow you to play around with traffic and shape it using the firewall-cmd command's different options.

Chapter Four

Securing Your Server

In the previous chapter, we looked at how you can secure the server using a firewall. We also looked at the importance of user accounts and other details. In this chapter, we will look at the different ways to secure a server without using a firewall.

Updating Servers Regularly

If you can access the server as the administrator, you need to update it regularly. To do this, run the following command: *apt-get update && apt-get upgrade*. It is important to ensure while you run the upgrade that you are on a VPS or virtual instance. This can damage the image and the server. So, you need to check if you can do this on your host.

Creating a Secondary User Account

You need to do this to ensure no user logs into the server using the root account. If you want to reduce the probability of unauthorized access on your server, you need to create a user and ensure you give them the relevant privileges to ensure they cannot perform unauthorized activities on the server. We have looked at how you can add new users to the servers. If you want the user to have access to the server with a specified set of privileges, you can give him sudo access.

You need to update these terms in the configuration file. Once you do this, you can save and close the file down. You need to ensure the user can access the server the way he needs to, and you can confirm this by logging into a second terminal on the same server. It is especially important for you to do this. Let us now remove the root users who access the server through SSH. If you want to accomplish this, you need to edit the configuration file: */etc/ssh/sshd_config*. It is important to ensure you are logged into the same server using a different shell before you reload the SSH. This is to ensure you

do not log yourself out of the server. Now, open the configuration file and update the line #PermitRootLogin yes to PermitRootLogin no. You can then restart the service using the following command:

```
/etc/init.d/sshd restart
Stopping sshd: [ OK ]
Starting sshd: [ OK ]
```

Bear in mind that Linux does not have a default root password when you install the operating system on your server. There is no root login, but you, as an administrator, will have sudo privileges and administrative access.

Setting up SSH Keys

An SSH key on your server allows you to connect securely to the server using an authorized key pair. This is an extra step you need to perform to ensure the server is secure from any unauthorized access. You need to log in to SSH using the root user account and run the following command: *ssh-keygen -t rsa -C "you@example.com"*. Once you click on accept, the default file locations and names will be added to the server along with the file names. You can re-enter the password for your user account and add the public key to the file on your server. Use the following lines of codes to do this:

```
cd ~/.ssh
cp id_rsa.pub authorized_keys
```

You should now copy the public key to the SSH directory in the root account that is present on the server. To do this, you can run the following commands:

```
cd ~/.ssh
scp authorized_keys root@host.servername.com:/root/.ssh/
```

You can now connect with the server directly.

Checking and Configuring the Firewall

Run the following commands to check and configure your firewall. You can also do it using the information present in the previous chapter.

```
root@server:~# ufw app list
```

Available applications:

```
OpenSSH
```

```
root@server:~# ufw status
Status: active
To Action From
-- ------ ----
22/tcp ALLOW Anywhere
22 ALLOW Anywhere
8080/tcp ALLOW Anywhere
80/tcp ALLOW Anywhere
Anywhere DENY 58.218.92.34
80 DENY 202.54.1.5
22 (v6) ALLOW Anywhere (v6)
22/tcp (v6) ALLOW Anywhere (v6)
8080/tcp (v6) ALLOW Anywhere (v6)
80/tcp (v6) ALLOW Anywhere (v6)
```

Limiting the Use of Open Ports

When you install Linux on your system, you do not have to worry about any network services listening to and monitoring the data you are passing through the server. When you start the server, you can determine who the administrator and root users should be and determine the various ports and services that can use the server. You need to test the open ports on your server. To do this, run the following lines of code:

```
root@server:~# /home# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address State PID/Program name
tcp        0 0 0.0.0.0:22              0.0.0.0:* LISTEN   69941/sshd
tcp6       0 0 :::22                  :::* LISTEN   69941/ss
```

Setting Up Live Kernel Patches

You can use the canonical livepatch service to determine when your server needs a security fix, especially when it comes to major kernel securities which do not reboot. As a Linux user, you can take advantage of these services for free up to three zones or nodes. Every machine that is a part of the subscription can find a way to fix security issues and vulnerabilities. You can learn more about these tools on the Linux website.

Hardening the Kernel

There are different built-in protections in a kernel, and you can use these methods to ensure the kernel is not compromised.

Hardening User Space

You need to learn more about how you can harden the user spaces. You can use different hardening tools and applications to do this. A common option is to use a compiler flag when you build a tool or application. This is especially true if you use the kernel to launch the application. You can add additional features to the application or tool if you need to. It is important to remember that these additional features can be added to all applications on the server, not just the operating system's official apps.

Using Secure Boot

When Linux developers launched Ubuntu, they included a secure boot to the operating system. The developers added the secure boot option, enforced it in the bootloader, and switched on the kernel's non-enforcing mode. With this setup, the versions that could not authenticate secure boot failed to boot, and the kernels did not validate the authentication, either.

Setting Up Two-Factor Authentication

If you want to add an additional protection layer to your server, you can set up multi-factor or two-factor authentication on the server. You need to be careful about setting this up, because you can lock yourself if you incorrectly enter the password. To do this, follow the steps below:

Step One

Access the server through SSH and run the following command to use Google as your authentication mechanism.

```
apt-get install libpam-google-authenticator
```

Step Two

You should now run the command google-authenticator to create a secret key in your server's home directory.

```
google-authenticator
```

Do you want authentication tokens to be time-based (y/n) y

Your new secret key is: 73GRSXVJNUXZWN2T

Your verification code is 389485

Your emergency scratch codes are:

99536578
44768915
90480600
82281337
56945099

Do you want me to update your "/root/.google_authenticator" file (y/n) y

Do you want to disallow multiple uses of the same authentication token? This restricts you to one login about every 30s, but it increases your chances to notice or even prevent man-in-the-middle attacks (y/n) y

By default, tokens are good for 30 seconds and in order to compensate for possible time-skew between the client and the server, we allow an extra token before and after the current time. If you experience problems with poor time synchronization, you can increase the window from its default size of 1:30min to about 4min. Do you want to do so (y/n) y

If the computer that you are logging into isn't hardened against brute-force login attempts, you can enable rate-limiting for the authentication module. By default, this limits attackers to no more than 3 login attempts every 30s. Do you want to enable rate-limiting (y/n) y

Step Three

You should now edit the configuration file for your SSH port. The file is called '*vim /etc/ssh/sshd_config*' and you can access it using PAM (pluggable authentication module). You need to run the following command before you save the changes and close the file:

```
UsePAM yes
ChallengeResponseAuthentication yes
```

Now, you can restart the server using the command below:

```
systemctl restart ssh
```

Step Four

Now that you have the SSH set up, you need to edit another configuration file where the PAM is present. This file is for the SSH daemon. You need to add the following command to the file 'vim /etc/pam.d/sshd': *auth required pam_google_authenticator.so*.

Step Five

You now have to save the file down. Close the file and reboot the server. You will now be authenticated using Google authenticator.

Turning Off Internet Protocols

It is important to identify which Internet protocol (IPv4 or IPv6) you use frequently. For instance, if you do not use the IPv6 protocol frequently, you need to disable it in the configuration file. Add the following lines of code to the configuration file to do this:

```
vim /etc/sysconfig/network
NETWORKING_IPV6=no
IPV6INIT=no
```

Understanding the Applications/Tools before Installation

Before you install an application or tool on your server, you need to see if any new software or tools are being added to the server along with the application. This will put your server at risk since it will add a newer opening to the server, leading to a vulnerability.

Removing Unnecessary Startup Processes

Linux has various startup processes which run in the system, and these are called targets. You need to look at the settings and see what you need to change or update before you make the change in the configuration file. You can set up servers to ensure they have only the required packages or servers set. This will reduce the amount of time you spend cleaning up your server space.

Reviewing Activities Regularly

Every file on Linux will be located on the server and the infrastructure. You can access the folder or directory with the backup. In this directory, you will have details of all the files on the server and the different logs and their details. You need to review the logs regularly to ensure no unauthorized activities are being performed on the server. If you want to look at the file, you can run the command: *less file.log*. You can use different commands to go through the logs in the directory.

Start Backing Up

In case of any disaster to the server or a hack, it is important for you to have a backup. A backup is your last line of defense, and it is recommended you follow the backup method 3 – 2 – 1. In this method, you need to do the following:

Step One: Maintain 3 copies of your data – one will be the primary source while the other two will be the backups

Step Two: You need to store the data on at least two storage media, such as tape drives, cloud, local drives, etc.

Step Three: There needs to be one copy of the data stored offsite

It is also important for organizations to account for these backups and ensure they have details of any failed backups. Linux is a mature system, and it is quite secure when compared to other operating systems. The operating system has an unparalleled ability to adapt to and set up different configurations. Therefore, it is one of the best systems to use if you want to have a self-managed and secure system. Linux also offers you to use a dedicated server option, and this is stable and fast.

Only Install the Things You Need

This is an important thing to bear in mind – do not download everything you find. You need to install the packages you want to use. Your server should never be overloaded. This means you should only install those packages, applications, and tools on your server if you need them. If you find any application, server, or package on your system that is not required, you need to remove it. If you have fewer packages on your server, you can avoid patching the code.

Use SELinux

SELinux is an enhancement you need to make to the kernel, and this uses a MAC or Mandator Access Control system to ensure any application or tool is defined based on a set of resources. If you want to install the package, use the following command: *apt-get install selinux-basics selinux-policy-default auditd*.

Once you download the package, you need to load the policy script. This is a modified version of the actual script and is included as part of the SELinux package. You should then move this script to the folder, */usr/share/initramfs-tools/scripts/init-bottom/*, and run the command written below: *update-initramfs -u*. You can use this to update the SELinux package. Once you do this, you need to run the following command to configure PAM, GRUB, and auto relabel creations: selinux-activate.

You should now reboot the server and ensure the changes you have made have taken effect on your server. It may take your system time for you to label the file systems when you boot the server. The system will reboot for the second time when it has labeled the files accurately. Now, run the command and ensure you have everything set up accurately on your server and system. Performing an audit by yourself will help you find different problems in the installation of SELinux. You can learn more about this using the Debian SELinux documents.

SELinux or Security-Enhanced Linux is a security mechanism provided by Linux developers which can be used in the kernel. SELinux provides three operations:

1. **Disabled**: When you use this mode, you can disable the use of SELinux on your server
2. **Permissive**: If you use this mode, you will only receive a warning or alert when unauthorized activity is performed on the server. It will also log the actions performed by the user
3. **Enforcing**: In this mode, the functions of SELinux are not limited, which means it looks at every aspect of the security of your server

You can manage the modes using the '*/etc/selinux/config*' file.

Securing the Console Access

It is important to protect a Linux server from end-to-end, which means you need to protect the server's access through a console. You need to disable the use of external devices, such as USB pens, CDs, or DVDs, once you set up the BIOS. You also need to set up the BIOS and use the grub boot loader

password to protect the updated settings.

Restricting the Use of Old Passwords

You can ensure users do not use old passwords when they are prompted to update their passwords. Your Linux server will save the old passwords in the file /etc/security/opasswd. You can ensure your server does this using the PAM module. If you are unsure of how to do this, follow the steps given below:

1. If you use CentOS, Fedora or RHEL, you can open the file */etc/pam.d/system-auth*. To do this, run the command: *# vi /etc/pam.d/system-auth*
2. If you use Ubuntu or Debian, you can open the file */etc/pam.d/common-password* using the command *# vi /etc/pam.d/common-password*
3. You need to add the following command to the auth section in your configuration file: *auth sufficient pam_unix.so likeauth nullok*

Once you do this, you need to add the following lines to the configuration file to ensure a user does not reuse his last few passwords. You can decide the number of passwords you want the server to store.

```
password    sufficient    pam_unix.so nullok use_authtok md5 shadow remember=5
```

This line will indicate to the server that it needs to remember the user's last five passwords. So, if the user enters one of the earlier passwords, he will receive an error from Linux asking him to choose a different password.

Checking Listening Ports

You need to look at the different ports in your server and view them along with the services they provide. To do this, you can run the following command: *netstat -tunlp*. Based on the list you receive, you can determine if there are any ports you need to disable from the system. You can do this using the command chkconfig. You can then close all the ports in the server which you do not need. To do this, you can run the command: *chkconfig*

serviceName off.

Disabling Login through the Root

As mentioned earlier, it is important to ensure that you do not log in to your server using the root account. Therefore, you need to disable the ssh port as the root account on the server's configuration file. Before you do this, let us look at an example where we create a user with sudo privileges, so you can use the ssh port to access the server and perform the necessary tasks. When you log in to the server, you can move from the user to the root account if you need to. Let us first create a new user. To do this, use the following command: *useradd user1*. You can now add a password to this user account using the command: *passwd user1*. Now that you have a user created on the server, you can give it sudo permissions using the following line: *echo 'user1 ALL=(ALL) ALL' >> /etc/sudoers*. You then need to use ssh to access the server using the new user account. This is one way to ensure you log in to the server.

Now, we need to disable the root access to the SSH port. This will mean no user can use SSH to log in to the server using the root account. You need to open the configuration file ‘nano /etc/ssh/sshd_conf’ to do this. You then need to update the line PermitRootLogin to no in the file. You then need to save this file and close it. Once you restart it, you will see that the settings have been updated.

Before you log out of the server, you should test if you can log in to the server using SSH using a user ID that was created earlier. To do this, you can open another instance in the terminal and log in to the council using SSH. If everything works as it should, you can log out of the server as the root user.

Change Ports

You can also change the default port used by SSH to access the network. You can add a layer of security to ensure your server is safe. To do this, you need to make updates to the */etc/ssh/sshd_config file*. In this file, you need to replace port number 22 with a number you want to use as the port. Once you save the file down, you can reboot the server. To do this, you need to use the following command: *service sshd restart*. You can then log in to the server using the updated port number with the following command: *ssh*

username@IP -p 1110.

Disabling Shortcuts

When you hit the shortcut Ctrl+Alt+Delete, your server will automatically reboot. Therefore, it is best for you to disable this shortcut to prevent someone from rebooting the system by mistake. This action is in the configuration file named */etc/init/control-alt-delete.conf*. All you need to do is to comment out the line '#start on control-alt-delete' in the configuration file.

Logging In Without Passwords

You can log in to your server using SSH. You can let go of passwords if you configure the details in the file. The only thing you need to do is to enter the server by generating a key using SSH. It is important to ensure the user can only enter the server from the machine where he generated the ssh key. Let us see how we can generate an SSH key to access the server: *ssh-keygen - t rsa*. Once you generate the key, you can add the key to the server you want to access using the following command: *cat ~/.ssh/id_rsa.pub*.

If you have more than one user on the server, you can give each user a ssh key to access the server from his system. You can do this using the following command:

```
cd /home/user1
ls -ll
```

To ensure that only some users have access to a ssh key for security reasons, you can configure the list of users in a ssh directory. You can do this using the following commands:

```
mkdir .ssh
cd /home/admin/.ssh
vim authorized_keys
```

You can also create a public SSH key and add it to the configuration file before changing the owner. To do this, run the following command: *chown user1 authorized_keys*.

If you do not want to use an SSH key, you need to disable the SSH login. To do this, you need to edit the configuration file using the following lines of

command:

```
Edit /etc/ssh/sshd_config
Passwordauthentication no
PermitRootLogin no
```

Once you do this, you can ensure only an authorized user can enter the server using the command: *ssh user-name@serverIP -p(port Number).*

Use fail2ban

If you want to use an application to see where a server is attacked frequently, you can use fail2ban. This application also tells you about any automated attacks that happen on the server. It also works on the firewall and updates it when there is any issue found in the server. Fail2ban changes the configuration in the firewall and will push it to block the IP address either for a specific period or permanently, depending on the type of attack.

Run the following command if you want to install fail2ban on your system: *$ sudo apt install fail2ban -y*. You should then copy the configuration file that is found on your system using the following command: *$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local*. If you do not have the jail.local file, you need to create it and then copy the content from the jail.config file to the jail.local file. You can use the following command:

```
cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
Edit /etc/fail2ban/jail.local file
```

You now need to make the required changes to the file:

```
[sshd]
enabled  = true
port     = ssh ( provide the port number if the default port is changed )
protocol = tcp
filter   = sshd
logpath  = /var/log/secure
maxretry = 3 ( max no. of tries after which the host should be banned)
```

findtime = 600 (You can use this parameter to set the window that the application will pay attention to when it looks for repetitive incorrect authentication in a few seconds)

```
bantime  = 600 (time duration for which the host is banned -in seconds)
```

Before you restart the application, you need to block the IP by setting the bantime parameter to -1. Once you do this, you can restart the application

using the following command: *$ sudo service fail2ban restart*.

It is as simple as this. The application will continue to examine the log files on the system and look for any attacks. The application will also have a list of all IP addresses, which it will prevent from accessing the server. If you want to view this list, use the following command: *$ sudo fail2ban-client status ssh*.

You can also use fail2ban to log in to SSH. You need to ensure the application is updated regularly, so you can access the server using SSH.

Creating a New Privileged Account

You should now create a user account. Bear in mind that you do not log in to the server using the root account. You can also create your own account. As mentioned earlier, you should create a sudo account using which you access the server. Now, log in to the server using the sudo account. We have looked at how you can create new accounts and give it sudo privilege access.

Uploading the SSH Key

When you create a new server, you need an SSH key to access the server. You can use a pre-generated SSH key if you want to or create a new key. To do this, run the following command: *$ ssh-copy-id <username>@ip_address*. You can now avoid the password when you log in to the new server using the SSH key.

Securing SSH

Once you do this, you need to make the following changes:

Step One: Remove the password authentication for the SSH protocol

Step Two: Do not allow users to access the root account remotely

Step Three: You also need to restrict access to IPv6 and IPv4

To do this, you need to make changes to the SSH configuration file. When you open the file, you will find the following lines of code:

```
PasswordAuthentication yes
```

PermitRootLogin yes

You need to change it, so the lines look at follows:

PasswordAuthentication no

PermitRootLogin no

You then need to restrict which IP the SSH service uses – either IPv4 or IPv6. You can do this by making changes to the AddressFamily option in the configuration file. If you want to change it, so it uses only IPv4, make the following change:

AddressFamily inet

It is only when you restart the SSH service that your changes will reflect on your server. Bear in mind that you should have two active server connections before you restart the SSH server. When you have an extra connection, you can fix any issue that may arise in case the restart does not go as planned. If you use Ubuntu, you can run the following command: *$ sudo service sshd restart*. If you use any Linux distros which has a Systemd configuration, run the following command: *$ sudo systemctl restart sshd*.

Creating a Firewall

In the previous chapter, we looked at how you can install, enable and configure a firewall on your system, so it allows only the network traffic you allow to access the server. You can also use an uncomplicated firewall on your server if you need to, depending on how well you code. If you cannot code, use the uncomplicated firewall. You can install it on your system using the following command: *$ sudo apt install ufw*.

The uncomplicated firewall, by default, will deny any incoming connections and allow everything to go out of the network. This means all the tools and applications on your system or server can connect to the Internet, but if anything external tries to connect to the server, it will not allow it. You need to ensure you can access HTTPS, HTTP, and SSH protocols on your server using the following commands:

```
$ sudo ufw allow ssh
$ sudo ufw allow http
$ sudo ufw allow https
```

After you do this, you should enable the uncomplicated firewall using the following command: *$ sudo ufw enable*. You also need to check which services and applications are allowed and denied by the uncomplicated firewall: *$ sudo ufw status*. If you do not want to use the uncomplicated firewall, you should type the following command: *$ sudo ufw disable*. Alternatively, you can use the firewall interface we discussed in the previous chapter, as well.

Removing Unused Network Services

Every Linux operating system has a server that has applications and servers that work with the network. You need to use most of these applications and servers, but you can get rid of some. Using the command ss, you can see the network services and applications being used on your Linux servers.

```
$ sudo ss -atpu
```

The output of this command will differ depending on the distros you are using on your system. The following is an example of the output you may see. The output shows that the Ngnix (ngnix) and SSH (sshd_ services are the only active services. These are listening to every activity on the server and are waiting to connect to another server.

```
tcp LISTEN 0 128 *:http *:* users:(("nginx",pid=22563,fd=7))
tcp LISTEN 0 128 *:ssh *:* users:(("sshd",pid=685,fd=3))
```

The process of removing an unused or dormant application or service will be different depending on the type of distros you are using. It also depends on the package manager on the operating system. If you use Ubuntu or Debian, you need to run the following command: *$ sudo apt purge <service_name>*. On CentOS or Red Hat systems, you need to run the command *$ sudo yum remove <service_name>*. If you run the ss -atup one more time, you can determine if the unused or dormant services are no longer running and are uninstalled on your operating system.

In this chapter, we looked at tips you need to keep in mind when it comes to hardening your Linux server without using a firewall. You can add additional security layers to the server depending on how you and your team use the system. Some of these layers include intrusion detection software, access control and management, two-factor authentication, individual application configuration, and more.

Chapter Five

Password Encryption Methods in Linux

Since passwords are the main security mechanism used these days, it is important to ensure that every user on the server or system has a difficult and secure password. Linux distributions come with password programs that ensure a user's password is strong. You can also use other encryption software available in the market if you want to add another layer of protection to the passwords stored in the server. You need to ensure the programs used to protect passwords are updated constantly.

You need to use encryption software since it is necessary now more than ever. There are different methods you can use to encrypt data, and every method has its own characteristics. We will discuss methods in the latter part of the chapter. Since Linux distributions use a one-way password encryption method, no user can learn another user's password. The method used is called Data Encryption Standard (DES).

The passwords, once encrypted, are stored in a file on the server. If you do not use shadow passwords, the passwords are stored in the '*/etc/passwd*' file. Otherwise, they are stored in the '*/etc/shadow*' file. When a user logs into the server or network, the password he enters will be encrypted and checked against the file where his password is stored. If the passwords match, the user can enter the system. Otherwise, his access is denied. He is given a number of attempts depending on what has been keyed into the server. The data encryption algorithm or standard works as a two-way algorithm that allows users to encode and decode any message by using the right keys. Most Linux distributions only use a one-way algorithm, which means no user can reverse the encryption to identify the password.

You can use different methods, such as brute force attacks or password injections, to guess a user's password unless the password is strong. Using

PAM modules, a user can encrypt his password using different routines in the module. If you are still unsure of the password being used, you can use “Crack” to assess your password's strength and the passwords of every other user on the system. Experts recommend that users run the “Crack” program on their server as often as possible to identify any insecure passwords. Only when you do this can you determine the strength of the passwords in your system.

Pretty Good Privacy (PGP) and Public-Key Cryptography

If you use public-key cryptography, there will be separate keys used to encrypt and decrypt the information passed through the server. Traditional forms of cryptography use one key to encrypt and decrypt information, and this key should be available to both parties. It is assumed that the key is transferred carefully between the users. The public-key cryptography method uses two different keys to prevent the loss of the key used to decrypt information during the transfer. A public key is available for anybody who performs the encryption. The user, however, must ensure the decryption key used by them is kept private. They should not share this key with anybody over the network.

PGP is another privacy option you can use to protect the data on your system and server. If you choose to use this, ensure you have the version applicable and allowed in your country. There are some export restrictions placed by the US government. Therefore, you cannot use strong encryptions when you transfer any information from outside the country.

S/MIME, SSL and S-HTTP

Most users are unaware of the differences between these forms of encryption and security protocols. They also do not know how they can be used. In this section, we will go through each protocol to know how to use them to secure your network and server.

S/MIME

S/MIME or Secure Multipurpose Internet Mail Extension is a standard encryption protocol. This is used to encrypt all messages passed through the Internet, especially electronic email. S/MIME is an open standard, and this was developed by three programmers - Ron Rivest, Adi Shamir, and Leonard

Adleman or RSA. Therefore, you will see it used in Linux quite often.

SSL

SSL or Secure Sockets Layer is an encryption method used often in Linux. It was developed to provide security when moving data across the Internet and was programmed by Netscape. SSL also uses different encryption protocols and ensures the client and server is authenticated. The protocol will operate at the transport layer. It will also create a secure encrypted channel within which the data is passed. This makes it easy for you to encrypt the data passed through the connection. This encryption method is used when you visit a secure website. For example, you may want to access an online document that is secured. To do this, you can use the communicator. On using this method, the SSL will create a secure communication network with the communicator using different encryption software.

S-HTTP

S-HTTP is another Internet protocol that is used to provide security when you pass data through the Internet. This protocol was first designed to provide confidentiality, integrity, and authentication when data is passed across the network. It also supports cryptographic algorithms and the use of multiple key-management systems through the option of negotiating between the different parties communicating over the Internet. This protocol is limited to the use of specific encryption software, and you can implement it to ensure every individual only accesses secure information.

Linux IPSEC Implementation

Along with Cryptographic IP Encapsulation (CIPE) and other forms of data encryption, there are different Internet Protocol Security (IPSEC) methods used in Linux. The Internet Engineering Task Force developed the IPSEC methods and tools to ensure the data passed between IP networks was safe. These methods and tools encrypt the data using different cryptographic programs. You can also use these methods to manage access control, confidentiality, integrity, and authentication.

The Linux FreeS or WAN IPSEC is a free implementation of IPSEC which can be used on almost any system. This service allows you to build a secure tunnel even when you use untrusted networks or connections. The IPSEC

gateway machine will encrypt all the information passing through the untrusted network. The machine also decrypts the information at the destination. The IPSEC gateway machine led to the development and use of Virtual Private Networks or VPNs to ensure there is a secure network used by machines, especially if they are connected to the same server or network from different locations and devices.

Secure Telnet (stelnet) and Secure Shell (ssh)

ssh and stelnet are a collection or suite of programs that enable a user to log in to the server or system remotely. These programs use an encrypted connection which allows them to log in safely.

Another suite of programs called openssh is used as a replacement for rcp, rlogin, and rsh. This is a secure alternative to these programs. This suite encrypts the communication or data passed between hosts through public-key cryptography. It also can be used to authenticate users on the server or system. This program can also be used by users to log in to the server remotely. Once they are logged in, users can copy the data between hosts and prevent DNS spoofing, man-in-the-middle, and other hacks. The suite compresses the data passed between users and secures the data passed between the connection.

SSLeay is another SSL protocol that has many features like stelnet. It was developed as part of Netscape's SSL protocol. It can be used on numerous databases and algorithms, including IDEA, DES, and Blowfish.

Pluggable Authentication Modules or PAM

New versions of Red Hat Linux and Debian Linux distributions use a unified authentication mechanism to move data between networks and other logical connections. This is termed as PAM or Pluggable Authentication Modules. These modules allow users to change their authentication methods and requirements as and when they need to. They can also use these modules to encapsulate any local authentication methods used without having to recompile the information or binaries. The following are functions you can perform using PAM:

1. Prevent Denial of Service (DoS) attacks from users by setting

resource limits for each user

2. Allow users to log in to the server only from specific devices and places at specific time intervals
3. Use shadow passwords (we will cover this shortly)
4. Use different forms of encryption, including DES, for passwords to ensure they cannot be detected through brute-force attacks

If you have this enabled in your server or network, you can prevent numerous attacks before they occur on your system.

CIPE or Cryptographic IP Encapsulation

The objective of this encryption is to provide a facility that is secure for the server and system. This encryption protects the interconnection between subnetworks across insecure packet networks from traffic analysis, eavesdropping, and faked message injection. Using CIPE, you can encrypt the data at both the server and network levels. All the information moving along the network in the form of packets is encrypted, and these packets are received and sent by an encryption engine that is next to the network driver.

CIPE does not encrypt the data that is passed through the network or connection. It does not even encrypt the data at the network or socket levels. It will only encrypt the logical connection between networks, sockets, and programs found on different hosts. This is very different from SSH. You can also use CIPE to create new VPNs through tunneling. If you want to avoid making changes to the application software being used, you can use low-level encryption methods. These methods ensure the logical connection between two networks connected in the VPN or Virtual Private Network work transparently.

Using Shadow Passwords

You can use shadow passwords to ensure a user's password is encrypted and the information is maintained in a file that normal users cannot access. New versions of RedHat and Debian Linus use this setting as a default. After encryption, the passwords are stored in a file in the server directory that every user on the server can access. If one were to run a password cracking or

guesser program on the file, they could obtain every user's password on the system.

A shadow password will be stored in a different file which can be accessed only be an administrator or privileged users. If you want to use shadow passwords, you need to ensure that every application, tool or utility which needs the password can access the information. You may also need to recompile the list of passwords to support the user's access to the system. Using PAM, you can plug in a shadow module that does not require you to recompile any of the system's passwords.

John the Ripper and Crack

If you use a password program on your server, but you notice that hard-to-guess passwords are not used, then you need to run a password cracking program. This program will help you determine if the passwords used by you and your users are safe and secure. The idea behind a password cracking program is simple – the program will look at every word in the dictionary and try variations of the same words. It will also encrypt the words and check those words against your password. If the program finds a match, it will know the user's password.

There are numerous password-cracking programs you can use, but people often use "John the Ripper" and "Crack." These programs may take up your CPU's time but give excellent results. Before a hacker uses these programs on your system and server, you need to use them and check if there are any weak passwords. If yes, ask the users to change them immediately to avoid any hacks. You may wonder how the hacker can access the password file on the server when he can only access the server through a hole or vulnerability. Unfortunately, there are holes in the server, and an experienced hacker can find those holes easily.

You can also try other encryption methods used in Linux, such as Transparent Cryptographic File System (TCFS), Cryptographic File system, display security, X11, and SVGA. We will look at them in detail later in the book.

Chapter Six

Tools to Encrypt and Decrypt Password Protected Files

The process of encoding or protecting files to ensure only selected users can access the file is termed encryption. We have been using the concept of encryption even before computers came into existence. When countries were at war, they developed codes to communicate with their spies or soldiers in the warzone. This was one way for them to ensure nobody else could understand the message that was being passed.

Linux distributions come with standard tools used to encrypt and decrypt the information in the system or server. In this chapter, we will look at seven tools you can use to encrypt and decrypt files using a password.

GNU Privacy Guard or GnuPG

GnuPG, often called GPG, is a collection or suite of various cryptographic software and is written in C language. If you want to use this to encrypt and decrypt files, it is best to download the file's latest version. Most Linux distributions released in the last few years will come with this package installed as a default. If you find that it is not installed in your server or system, use the following commands to install it:

```
$ sudo apt-get install gnupg
# yum install gnupg
```

Across all the examples in this chapter, we will use a file called tecmint.txt file to encrypt and decrypt. This file is on the desktop of the system and will be moved to a different location if required. Before we go ahead, let us look at the content in this file. To do this, run the following command: *$ cat ~/Desktop/Tecmint/tecmint.txt*.

You can now encrypt the tecmint.txt using the tool. When you run the GPC command using the ‘-c’ option. This is a symmetric cipher, and it will create

the encrypted file with the extension '.gpg' against the file tecmint.txt. You can look at the content in the directory to determine which file was encrypted:

```
$ gpg -c ~/Desktop/Tecmint/tecmint.txt
$ ls -l ~/Desktop/Tecmint
```

You need to enter the word 'paraphrase' twice if you want to encrypt any file. In the above example, we have used the encryption algorithm CA S T5 to encrypt the file. You can specify a different algorithm if you use a specific one often. If you want to look at the different algorithms available to you, enter the following command: *$ gpg –version*.

To decrypt the file we have encrypted above, you need to use the commands below. Before you do this, you need to remove the original file from the desktop and only leave the encrypted file. We do this since the decryption algorithm will replace the encrypted file with the original file, and you may receive an error if you have the older file on the desktop while you perform the decryption. The commands to run are:

```
$ rm ~/Desktop/Tecmint/tecmint.txt
$ gpg ~/Desktop/Tecmint/tecmint.txt.gpg
```

It is important for you to remember the password you use at the time of encryption. You need to use the same password if you want to decrypt the file.

Bcrypt

Bcrypt is a function based on the Blowfish cipher, and the developers tried to ensure that the algorithm used was not as weak as the Blowfish cipher. Most organizations stopped using the Blowfish cipher when they learned that users could attack the algorithm from anywhere because of a small vulnerability. To install the tool on your system, run the following commands:

```
$ sudo apt-get install bcrypt
# yum install bcrypt
```

When you encrypt the file, you need to run the following command: *$ bcrypt ~/Desktop/Tecmint/tecmint.txt*. When you run this command, the server will create a new file on your desktop and replace the original file. To decrypt the file, you need to run the command: *$ bcrypt tecmint.txt.bfe*.

Since bcrypt is not very secure when it comes to encryption, it cannot be used on some versions of Debian that do not have their own security layers.

Ccrypt

This tool was developed specifically for Linux and is like the crypt tool used for Unix. Ccrypt is a utility for the encryption and decryption of streams and files. The tool uses a cypher called Rijndael. You can use the following commands to install the tool or application on your system:

```
$ sudo apt-get install ccrypt
# yum install ccrypt
```

The tool uses the ccencrypt and ccdecrypt to encrypt and decrypt a file, respectively. It is important for you to notice that when you encrypt the file, you need to use the original file tecmint.txt. After encryption, you need to replace the file with the file tecmint.txt.cpt. When you decrypt the file, the original file is found in the folder instead of the encrypted file. You can use the 'Is' command to do this.

Use the following command when you want to encrypt a file: *$ ccencrypt ~/Desktop/Tecmint/tecmint.txt*. Similarly, the command used to decrypt a file is: *$ ccdecrypt ~/Desktop/Tecmint/tecmint.txt.cpt*. You must ensure to enter the same password when you encrypt and decrypt the file.

4-Zip

Users commonly use this tool, and it is the most famous archive format. Since this tool is used often, the files used are called archive files. If you have not installed this application or tool on your system, you can do it using yum or apt.

```
$ sudo apt-get install zip
# yum install zip
```

You can create an encrypted zip file by grouping multiple files together using the following command: *$ zip --password mypassword tecmint.zip tecmint.txt tecmint1.1txt tecmint2.txt*. Using the above code, you are using mypassword to encrypt the file you want to. The tool will create an archive with the name techmint.zip. There will be two zipped files, namely tecmint1.txt and tecmint2.txt, and you can use the following command to decrypt the file: *$*

unzip tecmint.zip. You need to provide the same password when you encrypt and decrypt the file.

Openssl

Openssl is a cryptographic tool used to encrypt and decrypt files and messages. This is run on the command prompt, and it's easy to use if you are familiar with coding. If you do not have the application installed on your system, you can do this using the following commands:

```
$ sudo apt-get install openssl
# yum install openssl
```

Let us first look at the commands you need to run to encrypt a file using this tool:

```
$ openssl enc -aes-256-cbc -in ~/Desktop/Tecmint/tecmint.txt -out
~/Desktop/Tecmint/tecmint.dat.
```

In the above command we used various options/keywords to encrypt the file. Let us look at what these terms mean:

1. Enc: encryption
2. -out: This is the path where the file will be decrypted
3. -in: This is the location where your encrypted file will be stored
4. -aes-256-cbc: This is the algorithm used to encrypt and decrypt the files

Now, let us look at how you can decrypt files using this tool. Run the following command in the command prompt:

```
$ openssl enc -aes-256-cbc -d -in ~/Desktop/Tecmint/tecmint.dat >
~/Desktop/Tecmint/tecmint1.txt
```

7-Zip

This is an open-source tool that is written in C++, and it can be used to compress and uncompress most information stored in files. If you do not have this tool on your system yet, you can either install it using the yum or apt commands as follows:

```
$ sudo apt-get install p7zip-full
```

```
# yum install p7zip-full
```

Once you do this, you need to compress the file into a zip folder using the tool and encrypt it using a password. Run the following command to encrypt the file: *$ 7za a -tzip -p -mem=AES256 tecmint.zip tecmint.txt tecmint1.txt*. If you want to decrypt the file, you need to run the command: *$ 7za e tecmint.zip*.

Bear in mind that you need to provide the same password when you encrypt and decrypt the files.

Nautilus Encryption Utility

So far, we have looked at tools that are based on commands. The Nautilus Encryption Utility is a graphic user interface tool. You can encrypt and decrypt files in the graphical interface.

Encryption

You need to follow the steps below if you want to encrypt the files using a Nautilus encryption utility.

Step One: Select the file you need to encrypt

Step Two: The next thing to do is encrypt the file into a zip folder and choose the location you want to save it to. You also need to choose the password you want to use to encrypt the file

Step Three: Once you do this, you will note that the encrypted file has been created in the location.

Decryption

Step One: Open the zip file created in the user interface. You will see a lock icon present next to the file's name. This icon indicates that you need to enter a password before you can view the contents in the file

Step Two: If you have entered the right password, the file will open for you

This is not an exhaustive list of tools you can use to protect files using encryption.

Chapter Seven

Using Tools to Encrypt Files on Linux

In this chapter, we will look at the different tools you can use to encrypt and mount various files and folders onto your partition.

Tomb

This is an open-source and free tool that can be used to encrypt data easily. You can also use it to back files on your Linux system easily. It consists of simple programs and shell scripts that you can use to implement in-built encryption tools on your systems, such as LUKS and cryptsetup. This tool aims to improve the safety of your data to ensure you encrypt the data without losing any data. It uses well-tested methods and standards. It is easy to implement. The tool comes with the right practices and methods to store data and mount encrypted data onto a file or partition.

Cryptmount

Cryptmount is an open-source and free tool that was created specifically for Linux operating systems. This tool allows you to mount any encrypted file in the partition created without using administrator or root privileges. The tool uses a devmapper mechanism or process, which offers numerous advantages. It also helps you improve the functionality of the kernel. It also supports the movement and partition of swaps for privileged users. It also supports the use of the crypto-swap mechanism when the system boots. You can also store multiple encrypted systems and files onto one disk.

CryFS

CryFS is an open-source and free cloud-based encryption application that allows you to encrypt and store files anywhere on the cloud. It is extremely easy for you to set up, and you can run it in the background. It works with all the popular cloud services, including iCloud, Dropbox, and OneDrive. The

tool ensures there is no data anywhere but in your system and on the cloud. This data includes metadata, file content, and directory structure.

GnuPG

GNU Privacy Cloud, GnuPG or GPG is a free and open-source tool you can use to encrypt files on the server. It comes with a collection of cryptographic tools and you can use this tool as a replacement for Symantec's PGP cryptographic software. This tool or application is compliant with the various standards and tracks specific to the RFC 4889 and OpenPGP. We have looked at how this can be used in the previous chapter.

VeraCrypt

This is an open-source, free and multi-platform tool which can be used to provide any user with the option to encrypt the files whenever they need to. You can use the tool to encrypt selected partitions or an entire device using different methods of authentication. VeraCrypt allows users to:

Step One: Create disks, encrypt them and mount them onto the shell

Step Two: Ensure the files appear like they are real

Step Three: Pipeline

Step Four: Parallelize the drives and storage units into different partitions

EncFS

This is another open-source and free tool that is used on Windows and Mac to mount EncFS folders. You can also use it to edit, create, export, and change the encrypt EncFS folders' password. If you wish to use this tool, you need to download the latest version. It is also compatible with Linux.

7-zip

This is a free and popular tool and is open-source to an extent. You may have to pay to use some functionalities in the tool. It is a multi-source platform and uses a file archiving mechanism to compress files or directories into containers. These containers are termed as archives. 7-zip is a popular utility tool used because of the compression ratio using the forms LZMA and

LZMA2 with the 7z format. It also comes with a plugin for a FAR manager and can be integrated with Windows. Some other features of 7-zip are AES-256 encryption.

Dm-crypt

Dm-crypt is another tool used to encrypt the files and folders stored at the disk level. It does this by encrypting portable containers, disks and partitions. This tool was developed to address reliability and vulnerability issues found in the tool cryptoloop. It can also be used to back up large volumes of data of different forms.

eCryptfs

eCryptfs is an open-source and free collection of software and tools you can use to encrypt data on disks used in Linux. This tool works like the GnuPG tool, and it implements a POSIX-compliant layer of encryption on the files and systems. This is a part of the Linux kernel and comes installed with most Linux distros. Every version after the Linux 2.6.19 version comes with this tool. eCryptfs is easy for beginners to use since it helps you encrypt partitions and directories without worrying about the file system.

Cryptsetup

Cryptosetup is a tool or application used to enable users to encrypt various files using a DMCrypt, a kernel module with an emphasis on the Linux Unified Key Setup (LUKS) design. This key has become a standard encryption technique used in the Linux hard drive because it is compatible with every distro available on Linux. This design also ensures that data can be migrated or transported through the network smoothly by securing the information through passwords and other encryption methods.

Chapter Eight

Using Cryptsetup to Setup Encrypted Filesystems and Swap Space

A Linux Foundation Certified Engineer or LFCE is trained and has the knowledge to handle Linux network services. He will be trained to install, troubleshoot and manage network services in these systems. He also deals with the maintenance, design, and implementation of the architecture used in Linux systems. If you are a certified LFCE, you can do everything out there to manage the data and files stored in the system or network.

The objective of encryption is to ensure you have only trusted people accessing your information, especially sensitive data. This is the best way to protect the data from being used by hackers in case of any data leak. You can also use this method to protect the data from being stolen or lost from your hard disk or machine.

In simple words, the objective of encryption is for you to use a lock to access the information on your system or server. This is done to ensure the information only is available when you run the system. You can ensure any unauthorized users cannot unlock the system. This means if a person tries to look at the content on the disk, he will not find any information on the files.

In this chapter, we will look at how you can use the dm-crypt to set up an encrypted file system in your server. The term dm-crypt is short for device-mapper and cryptographic. This is the standard tool used for encrypting data in the kernel. It is important to note that the dm-crypt tool can only be used at a block-level. You can use it to encrypt the entire device, loop device, or partition.

Using a Drive, Loop Device, or Partition for Encryption

Before you perform encryption on your drive, you need to backup all the files on the drive, partition, or loop device before you wipe the data in the chosen

space. It is important for you to do this before you proceed further. You also need to wipe the data from the device used, and to do this, we will use the dd command. You can also perform these actions using different tools, such as shred. We will now create a partition on the device, /dev/sdb1, using the command # dd if=/dev/urandom of=/dev/sdb bs=4096.

Testing the Encryption

Before we see how we can use cryptsetup to encrypt files, we need to ensure the kernel is compiled of the encryption you use. To do this, you can write the following command: # grep -i config_dm_crypt /boot/config-$(uname -r). You need to ensure the kernel module dm-crypt is set up correctly to ensure the files and systems can be encrypted in the server and network.

Installing cryptsetup

Cryptsetup is a tool used to create, configure, access and manage any encrypted files on the server or network. You can do this using dm-crypt. Run the following commands to install the tool on your system:

```
# aptitude update && aptitude install cryptsetup        [On Ubuntu]
# yum update && yum install cryptsetup                          [On CentOS]
# zypper refresh && zypper install cryptsetup          [On openSUSE]
```

Setting the Encrypted Partition

The cryptsetup using Linux Unified Key Setup (LUKS) as the default operating mode, so this means you can stick to it. You can set the partition along with the passphrase using the command: # cryptsetup -y luksFormat /dev/sdb1

In the above command, you can run the cryptsetup using some of the default parameters. If you do not know what these parameters are, you can list them using the command: # cryptsetup –version. Using the list of parameters on your screen, you can determine if you want to change the key, cipher, or hash parameters. If you want to change them, you need to add the symbol ‘-‘ before the parameter and add a flag to it. You can then change the parameter which you take from the file /proc/crypto. You then need to open the partition you have created for LUKS. You will be asked to enter the passphrase you used previously. If your authentication passes, you will see that the partition is encrypted in the server or system. This will be present inside /dev/mapper

using the following name: # cryptsetup luksOpen /dev/sdb1 my_encrypted_partition.

We will now save the partition using the format ext4. To do this, run the following command: # mkfs.ext4 /dev/mapper/my_encrypted_partition. This will allow you to create a mount point in your partition, which will help you mount the partition encrypted in the partition. You can also use the commands below to determine if the mount operation is successful:

```
# mkdir /mnt/enc
# mount /dev/mapper/my_encrypted_partition /mnt/enc
# mount | grep partition
```

If you are done reading from or writing to the file system you have encrypted, you need to unmount it using the following command: # umount /mnt/enc. Once you do this, you can close the LUKS partition using the following command: # cryptesetup luksClose my_encrypted_partition.

Testing Encryption

Now that you have encrypted the file, you need to check if the partition or swap you have encrypted is safe.

Step One: The first thing you need to do is enter the LUKS partition you have created. To do this, run the command: # cryptsetup luksOpen /dev/sdb1 my_encrypted_partition

Step Two: Now, enter the passphrase you entered to encrypt the file

Step Three: The next thing to do is to mount the partition using the following command: # mount /dev/mapper/my_encrypted_partition /mnt/enc

Step Four: You need to create a dummy file in the mount point using the command: # echo “This is Part 3 of a 12-article series about the LFCE certification” > /mnt/enc/testfile.txt

Step Five: You also need to verify if you can access and use the files you have created in the mount point using the command: # cat /mnt/enc/testfile.txt

Step Six: To unmount the system, you need to enter the command: # umount /mnt/enc

Step Seven: Now that this is done, you can close the partition using the

command: # cryptsetup luksClose my_encrypted_partition

Step Eight: When you try to mount the file in the regular file system, it will indicate an error. If it does, you have done the right thing to encrypt the partition

Adding Additional Layers of Security

You can also encrypt the swap space in the server to add another layer of security. When you enter a passphrase on your server to encrypt the partition in the server, it will be stored in the RAM when you switch your system on. If any hacker wants to use this key, he can easily decrypt this information in the data. It is especially easy to do this when it comes to a laptop since the partitions of a RAM are maintained, even during their hibernation sessions, in the swap layers or sections of the memory.

If you want to avoid leaving a copy of the accessible to a hacker, you need to encrypt this swap layer or partition using the process mentioned below:

Step One: Create a partition or separation in the RAM, which will be used as the swap. You need to maintain the right size and use the encryption mechanism mentioned earlier. You can name this section as swap if you need to

Step Two: Once you set this partition, you need to activate the swap using the following commands:

```
# mkswap /dev/mapper/swap
# swapon /dev/mapper/swap
```

Step Three: You should now change the entry of the swap in the file: /etc/fstab. To do this, use the following command: /dev/mapper/swap none swap sw 0 0

Step Four: The last thing you need to do is to save the file down and reboot the server. To do this, run the following command: swap /dev/sdd1 /dev/urandom swap

When you reboot the system, you can verify if the swap space is active or not using the following command: # cryptsetup status swap.

Chapter Nine

Using Access Control Lists in Linux

Now that you know how to encrypt and protect your files and data using a password, let us understand how you can control access to the servers in Linux.

Introduction to Access Control Lists (ACL)

An ACL is one that provides flexible and additional permissions for your files and directories. It is designed so you can work with the different file permissions in Linux. ACL also allows you to identify the permissions you need to set for each user or a group of users. You can also use it to protect any disc resource.

Uses of ACL

Consider a situation where you have a user who is not a part of any user group on the server or network. This user needs to access files and make changes to them, so you will need to give him access using commands. This is where the concept of Access Control Lists comes into the picture. Using ACLs, you can grant flexible permissions to specific users in the system. If you are an administrator, you can use an ACL to define the fine-grained access rights for directories and files. You can use the keywords setfacl and getfacl to set up an ACL and view the ACL on your server or system. For instance, if you run the following command: getfacl test/declarations.h, you will receive the output below:

```
# file: test/declarations.h
# owner: mandeep
# group: mandeep
user::rw-
group::rw-
other::r—
```

List of Commands to Set Up ACLs

In this section, we will look at the different commands you can use to set up ACLs in your system.

Adding Permissions to Users

You can do this using the following command:

```
setfacl -m "u:user:permissions" /path/to/file
```

Adding Permissions to Groups

To do this, run the command: setfacl -m "g:group:permissions" /path/to/file

Allowing Files and Directories to Inherit ACL Entries

You can use the following command to allow a file or directory to inherit the properties or privileges found in an ACL from the directory it is a part of: setfacl -dm "entry" /path/to/dir.

Removing a Specific Entry in the ACL

To do this, you can run the command: setfacl -x "entry" /path/to/file

Removing Entries in ACL

There may be times when you might want to remove the entries in the ACL. To do this, you need to run the command: setfacl -b path/to/file. For instance, you can use the following to remove the declarations against the user Mandeep: setfacl -m u:mandeep:rwx test/declarations.h

Modifying the ACL

Adding Permissions for Users

To do this, you need to enter either the username or ID of the user who you want to add to the list. Use the following command to do this: # setfacl -m "u:user:permissions".

Adding Permissions to Groups

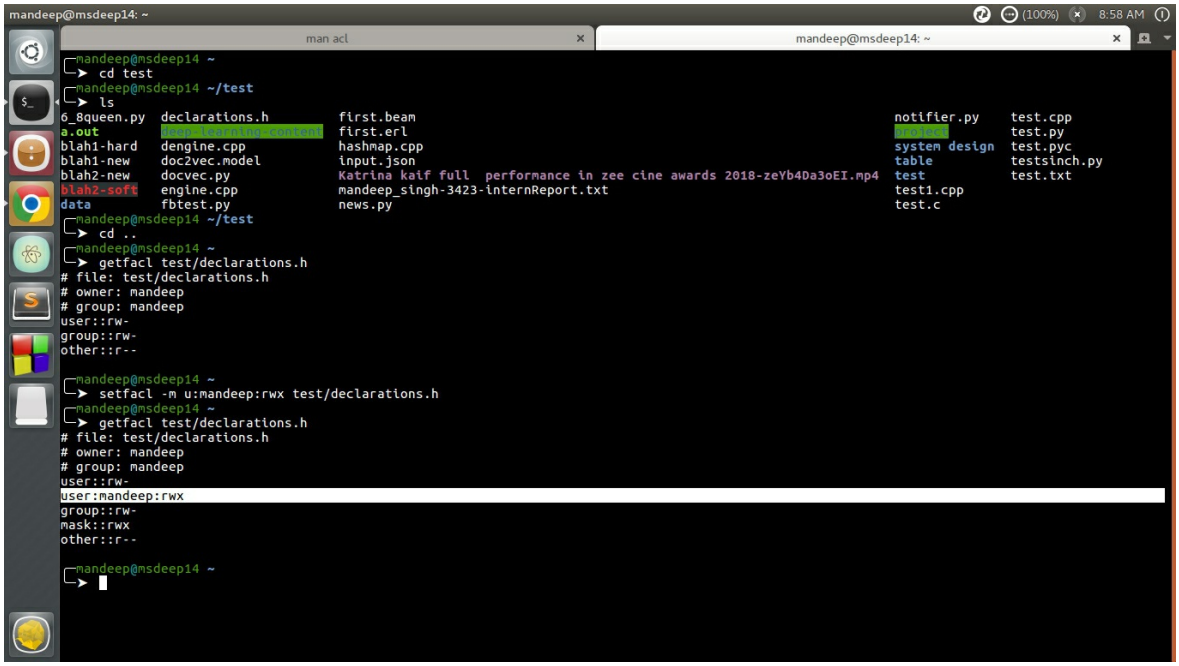
This is similar to what was done in the previous section. You can use the group name and ID to add permissions. Use the following command to do this: # setfacl -m "g:group:permissions"

Allow Files or Directories to Inherit the ACL Entries

You can use commands to let the files and directories obtain the properties

from the directory where the files and directories are present. To do this, you need to use the command: # setfacl -dm "entry".

For example, when you run the command setfacl -m u:mandeep:r-x test/declarations.h, the output will be the one in the image below.

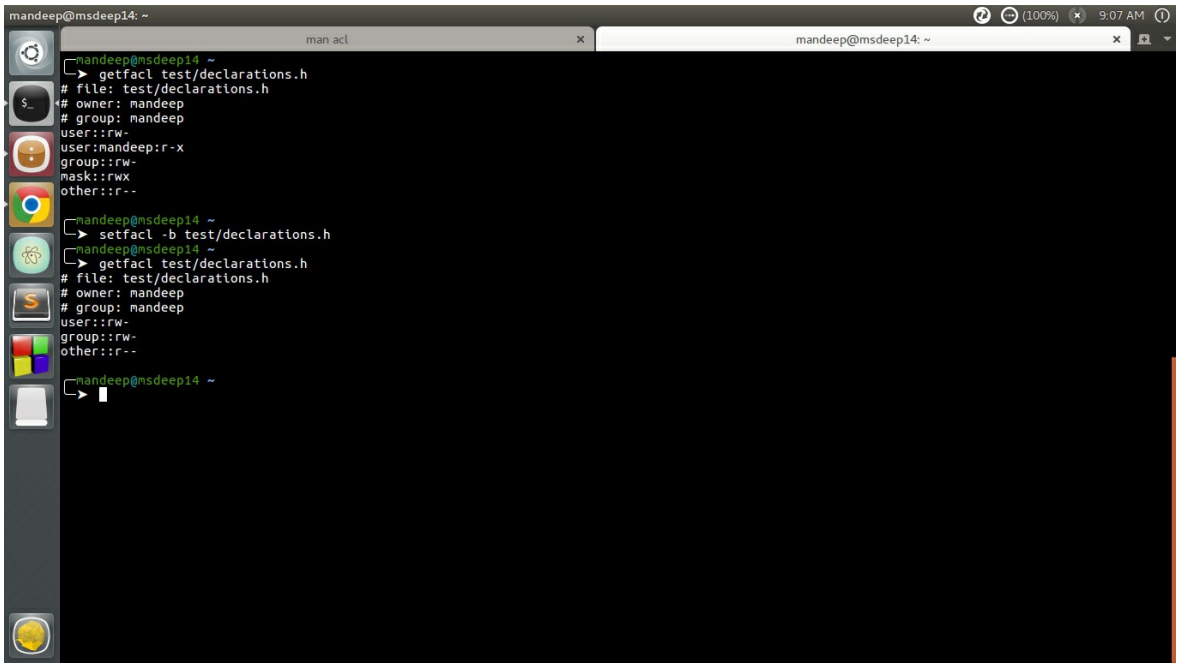


Viewing ACL

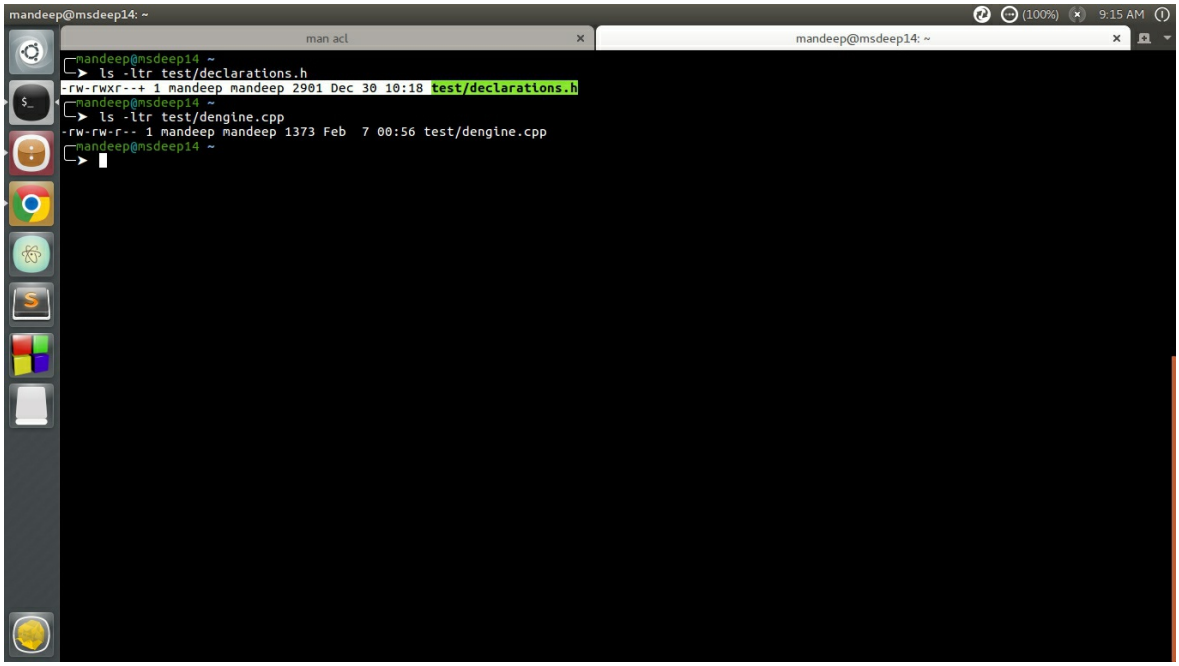
You can use the following command to view the permissions written in ACL: # getfacl filename. The differences between the getfacl and setfacl commands are quite obvious. So, pay attention to them. We have used an extra line against the username Mandeep, and this is in the image above. You can use the above command to change the permissions in the ACL from rwx to r-x.

Removing ACL

If you want to remove any commands you set in the ACL, use the setfacl command with the option ‘-b.’ Consider the example below:



Now, compare the output when you run the getfacl command before and after you use the setfacl commands with the option '-b.' You will see there is no entry in the list with the name mandeep in the second output. You can also look for any extra permissions which were set through the ACL using the command 'ls.'



When you look at the first command in the above image, there is a plus sign immediately at the end of the ACL's permissions. This is an indication that

there are some extra permissions that you can set in the ACL, and you can do this using the getfacl command.

Using Default ACLs

The default ACL uses a specific permission type that is assigned to the directory where you are making changes directly to the directory. This makes it easier for you to use the ACLs which come with your Linux. Let us see how you can use it. We are now going to create a directory or file and assign the default ACL to the file or directory using the option ‘-d’ using the following command: $ mkdir test && setfacl -d -m u:dummy:rw test.

You now have an idea of what you can do to maintain security in your systems and server. The next two chapters shed light on how you can perform penetration testing and scanning to find any vulnerabilities in your server and network.

Chapter Ten

Downloading and Installing Kali Linux

If you want to perform an analysis of your system's security, you need to use the Kali Linux distro. This is a powerful tool you can use to assess the vulnerabilities in the system. In this chapter, we will look at how you can download and install the Kali Linux distro in your system and use various penetration testing tools which are built into the operating system. These tools will help any system administrator while conducting penetration tests in the various stages of the penetration testing lifecycle.

We have covered the details of how to install an operating system onto your system using a virtual machine, but let us look at what you should do when you install the operating system directly onto your system. For those of you who have never installed an operating system ever, there is nothing to worry about, as this chapter will guide you with a detailed installation of the Kali Linux operating system. From the information in this chapter, you will learn how to install Kali Linux and where you can download the installation media from.

Kali Linux is a tool that you can use to check the system for any vulnerabilities. This operating system installs quickly on permanent media like a hard disk but can also be installed on a USB stick and live booted from it whenever required. This is a very convenient and portable tool in the toolkit of a system administrator. If you can access the local machine while you work as a system administrator, you can leverage the Kali Linux live disk to boot it into a locally available physical machine inside the target organization's infrastructure. There are over 400 tools and applications available in the Kali Linux operating system.

Downloading Kali Linux

Kali Linux is a distribution of the Linux operating system, and we discussed the same in the first chapter. This operating system is available for free, and

you can download the ISO image file from the official website. You may need to use a different system to download the ISO and then burn it onto a DVD or USB to ensure you install it onto your system. You can download the image from the following URL: https://www.kali.org/downloads/

If you need material to learn more about the configurations, advanced operations, and other special cases, you can find this information on the official Kali Linux website at

http://www.kali.org/official-documentation/

It is recommended that you register on the Kali Linux website to access the community where you can discuss your issues and learn more about how to overcome them.

Before you download the image file from the website, you need to ensure you select the right architecture. Your system architecture is 32-bit or 64-bit, and you need to choose the image which works for your system. Once you complete the download, you can use the image burning software to copy the Kali Linux installation media onto a DVD or USB stick.

In this chapter, we will look at how you can install Kali Linux using a USB drive and Hard disk using Live boots.

Hard Disk Installation

Once you start with the installation process, you need to place the DVD in the drive and plug the USB stick to install the media used to load the operating system. Depending on what you want to use, you should set up the boot priority in your computer's BIOS settings, so this installation drives through the right media.

Booting Kali Linux for the First Time

If you have managed to install the Kali Linux exe file from a USB stick or DVD, you will be given options on the screen about any existing information about the operating system and any other operating system on your hard disk. It will also replace the files with Kali Linux. Other advanced options are using which you can load the operating system on part of your hard disk using your existing operating system. This is beyond the scope of the book. Let us now look at how you can install the operating system.

Setting the Defaults

The next few screens will have the default settings you need to use when you set up the Kali Linux operating system. You can choose various options, such as the language, location, and language for your keyboard. You need to select the settings which you need to apply to the region and device. You also need to click on the next option to proceed with the installation process. You will be given information about the progress of your installation.

Initial Network Setup

A screen will appear on your system, and you can use it to type the hostname. Ensure the word is unique. When you click next, the installation manager will request you to enter the fully qualified domain name. The installation manager will use this domain name to download and install Kali Linux on your network or server. You can skip this step if you need to while you install Kali Linux and run it as a standalone system. If you are unsure about what you need to do, you can leave the option blank and continue.

Password

On this screen, the installation manager will ask you to choose the password for the root account. The root account is a super or privileged account, and it can be used to own the system. The Kali Linux system comes with a default password for the root account, and this is ‘toor.’ It is recommended for you to change the password to the complexity of your choice. You also need to ensure the password you enter adheres to the following criteria: uppercase, lowercase, number, and symbol. Ensure to set up a complex password to secure your system from getting accessed by the wrong hands. Once you choose the password, you need to click on continue to move to the next step in the installation process.

System Clock

Another screen prompt will appear on your screen, where you can choose to set the alarm clock. Now, select the time zone and continue with the installation.

Disk Partitioning

There are different methods you can implement when it comes to using partitions when you install Kali Linux using a hard disk. There is a lot of information one needs to know when it comes to disk partitioning. In this

chapter, we will only look at the basic process that is called Guided Partitioning. Since we are using a Guided partition mechanism, you need to use the entire disk to install the operating system. Select the option and continue the process of installation.

On the next screen, you will obtain information about all the hard drives which are on your system. You will see only one hard drive in the list and not multiple drives since most systems only come with one hard drive. Choose the hard drive you want to use to install Kali Linux. Now, select Continue. On the next screen, the installer will prompt you to determine how you will use the available hard drive.

I am assuming you are a beginner reading this book and will guide you on how you should do this. You need to select the 'All Files' option and choose one partition. This will keep the process extremely simple. Once you select this, click continue. In the next section, you will be given all the options which you can use as an input. You will be given the details selected to review. If you are happy with the details you have provided, you can continue with the information. There is only going to be one primary partition where all the files and folders will be saved. The secondary partition is called a swap. The latter is used as the virtual memory system where you switch between the RAM and CPU and the files are moved between these two locations. In simple terms, this is termed as buffer memory.

It is important to ensure all the swap partitions are only on Linux systems. This is the best way to encrypt the swap and protect the information in the swap. The swap needs to be the same size as the RAM that is installed on your system. Once this is done, you can select finish and move onto the next step. Once you are done with the selection, you can confirm the options selected by you. You will also be presented with a prompt to select yes and continue with the installation. You can change the partitioning scheme if you need to, but you can only do this when your operating system is live. This will damage your system. After clicking Continue, you will see a progress bar screen with the, and the installer will begin copying files to your hard disk. The time your system takes to complete the installation process is dependent on the hardware in your system.

Configuring the Packet Manager

When you finish copying the files onto the hard disk, you will receive a prompt that will ask you how you would like to configure the packet manager for your system. This package manager is especially important for your system, and it is used by the Kali Linux system when it needs to update the package repository based on the new updates made to the software. You need to use a network that mirrors the one that Kali Linux is sitting on since this is the only way you will learn about the package sources available for Kali Linux.

You now need to click on yes to continue the process of installation. You will also be prompted with another screen to specify the use of a third-party URL network package. This is again used when your Kali Linux system is part of a corporate system that stores a local repository for Kali Linux packages on its local server. If you do not want to fill in these details, you can leave it blank and continue with the installation process.

Installing the GRUB Loader

When you move onto the next screen, the installer will throw a prompt asking you to choose if you wish to install the GRUB bootloader when you run the Kali Linux system on your computer. The Grand Unified Bootloader, or GRUB, is the main screen that will appear on your screen when the Kali Linux operating system will boot up. The GRUB gives you a menu that you can use to move into the advanced settings before you boot the Kali Linux on your computer. You do not have to do this if you are an advanced user, but you need to install it if you are a beginner. Select Yes and click on Continue.

Completing the Installation

You are now at the end of the installation phase and will find the system's completion screen. Once this is done, you can click on Continue, and your system will automatically reboot. You can remove the installation DVD or use a USB stick if you need to before you reboot your system. At this point, you will see the Kali Linux welcome screen on your system. You need to log in using the root account you had initially set up and finally access Kali Linux.

USB Drive Installation

You can use a USB drive, thumb drive, or stick to install Kali Linux. This is a

storage device, and you can plug this into the port of your system. It is recommended for you to use a USB drive with at least 10 GB storage space or more if you want to install Kali Linux on your system. Every new computer system will allow you to boot the Kali Linux operating system from a USB device. You can choose to set the option to boot priority for your USB device from the BIOS settings on your system. We will look at how you can use a USB drive to install Kali Linux on your Windows or Linux machine. You can look at the documentation provided on the Kali Linux website to understand this better.

When you use a USB drive, you need to remember two terms – persistence and non-persistence. The former is the system's ability to retain the modifications or changes you make to the file even after you reboot the system. The latter refers to the fact that your system will lose its changes when you reboot the system. In this chapter, we will look at the persistent and non-persistent installation of Kali Linux on Linux and Windows operating systems, respectively. This way, you will learn both methods.

Windows Non-Persistent Installation

Before you move with the installation of Kali Linux using a USB drive through your Windows operating system, you need to download the Win32 Disk Imager from the following URL: https://sourceforge.net/projects/win32diskimager/

Once you download the ISO for Kali Linux like you did when you used the hard drive to install it, you can plug your USB drive into the system. Your current operating system will detect this automatically, and it will assign a driver to the file. You then need to launch a Win32 Disk Imager application and then click on the folder to browse through the different files and folders. You need to select the ISO for Kali Linux and download the file and click on the OK button. Select the drive letter you want to assign to the USB drive from the drop-down. You now need to click the write button, so your operating system will write the Kali Linux operating system onto your USB drive.

This process will take time, and this is dependent on your system's hardware. When the installer has completed moving the ISO to the USB drive, you need to restart your computer. Set the option to highest boot priority for the USB

drive form the BIOS settings in your system. It is important to understand that each system has its own user interface for the BIOS settings depending on the manufacturer. Therefore, you need to select the boot priority settings carefully. Once you do this, you need to reboot the system and the first thing on your screen will be the Kali Linux boot menu. You can choose the Live option that is the first option when it comes to booting Linux from the USB drive directly.

Linux Persistent Installation

It is important to remember that the size of the memory you have matters, especially if you use a USB drive to install Kali Linux on your system. Depending on the type of operating system you use, you need to create a Kali Linux USB drive. You need to ensure that the GParted application is installed on your device before you work on installing Kali Linux. If you have trouble with installed GParted, you need to read the documentation and see what you should do to get rid of any errors during installation. You can use any of the following commands in the terminal to install this tool:

```
apt-get install gparted
aptitude install gparted
yum install gparted
```

Once you have GParted downloaded on your system, you need to download the Kali Linux ISO from the official website and plug the USB drive onto your computer system. You need to use the command below in the Linux terminal if you want to see where the USB drive is present.

```
mount | grep -i udisks | awk '{print $1}'
```

The file in the USB drive will be saved using the name /dev/sdb1. You must ensure you look at the right file when you install Linux on your system since the file can be different from one system to the next. In the following command, you can remove the numbers at the end to ensure you switch between file names.

Use the dd command to write the Kali Linux ISO to the USB drive as follows.

```
dd if=kali_linux_image.iso of=/dev/sdb bs=12k
```

Launch Gparted application using the following command.

```
gparted /dev/sdb
```

You should now have one partition in your drive which has the Kali Linux image on it. The next thing to do is create another partition in the drive by right-clicking on the drive and selecting the ‘New’ option from the menu. This will appear once you select the partition menu available in the Menu bar. The following are the steps you need to follow, and they may vary depending on the machine you use:

- The unallocated space in the drive is greyed out
- You should now select the new option from the partition using the drop-down menu
- You can specify the size of the drive using a graphical slider
- Now, choose the file system and update it to ext4
- Select the add option and click on the drop-down menu. You need to select the option to apply all the operations on your system
- When you see a prompt, click ok, and the settings will update

You can add a persistence function to the USB drive using the following commands.

```
mkdir /mnt/usb
mount /dev/sdb2 /mnt/usb
echo "/ union" .. /mnt/usb/persistence.conf
umount /mnt/usb
```

That is it. You have now created a persistent Live Kali Linux USB. Reboot your system and you should be able to boot the Kali Linux operating system from the USB drive.

Chapter Eleven

The Penetration Testing Life Cycle

As a system administrator, you need to perform penetration testing in your organization to ensure you have the right security enabled in your network and server. In this chapter, we will look at the different stages used in penetration testing. Using this chapter, you will learn more about system administrators' different tools and the processes they need to follow during each stage. This way, you will understand the five stages of the penetration testing lifecycle and have an idea of the tools used by security professionals while engaging in penetration testing.

The Five Stages of the Penetration Testing Life Cycle

In this section, we will look at the five stages in the penetration testing lifestyle, and the information is broken down easily to help you understand what exactly is expected out of you during each stage.

Stage 1: Reconnaissance

This is the first step in the penetration testing phase. Let us assume you are running a military operation. The officers and analysts are going through the foreign territory map, and others in a different room are looking at the news and trying to see what can be done to take down the enemy. Another group is working on the information they need to collect to ensure the team can move into the enemy territory and make it their own. You need to do this when you work on a penetration test during your ethical hacking process.

If you do not have the expertise to run a penetration test, the organization will hire a team to perform this test. During this phase, you need to discover the information about the target system and also gather information about the system using public and private sources. You can do this easily by looking for information about the Internet since most information is publicly available. During this stage, you need only to understand what is present in the target’s network and server. You do not breach the network and server

since you only have to scan and document the information you will need while performing the next few steps.

Stage 2: Scanning

Let us consider that you are on a military mission with a team of people, and there is a hilltop you need to reach. This hilltop is behind your enemy lines, and one of your soldiers is hidden in the bushes. This soldier will come back and give you information about the enemy camp. The objective of this mission is for you to determine the type of activities the enemy is performing. This soldier will also tell you about the different ways you can move into enemy land and the security that is around the camp.

This soldier was told what he needed to do for him to get the information he needed. He was told what he had to look at. This is exactly what you do in the reconnaissance stage. You gather information, and it is this information you use during the scanning stage to determine which server and network you need to scan to find any vulnerabilities. You can use various tools to scan the target network, and Kali Linux comes with in-built options. You need to use the information you collect during this stage during the exploitation phase to determine which areas you need to exploit so that you can extract information.

Stage 3: Exploitation

If you are performing the penetration testing with a team, you need to ensure you move through the network and server easily without being detected. You also need to find a way to enter the server without getting caught. You need to find the right gaps in the server and enter the system and network through the server's open door. You also need to spend time inside the server and see what information you can extract from the system. You need to do this during the exploitation stage. The objective is to ensure you quickly enter and exit the network and server with the information you need without being detected by anybody. During this stage, you need to exploit the system and find sensitive information about the business. You can do this repeatedly if you need to.

Stage 4: Maintaining Access

When you perform the penetration test, you must ensure you have unlimited access to the servers and network you are using. You need to ensure there is

access to all the information you need. As the system administrator, you need to find a way to keep access while you discover how to enter the target system and network. You also need to find a way to exploit the server and see what information you can obtain or access through the network. You also need to find a way to get in and out of the system without getting caught. Therefore, with the information gathered in the exploitation state, they automate a way to keep their access continued to the target system.

Stage 5: Reporting

When you perform penetration testing on your system and server, you need to stand in front of the entire team and the higher-ups to tell them what you learned from the penetration test you performed. You need to explain every step and detail everything you found during the test. You need to expand the details of your findings. At the end of the penetration testing lifecycle, you also need to prepare a report which explains each phase of your penetration test. You need to explain the loopholes discovered, the vulnerabilities exploited and the servers, networks and systems targeted during the test. In some cases, you may need to provide information about the findings. You need to ensure management will look at the different means they can use to protect their servers and networks.

Chapter Twelve

Scanning

In this chapter, you will learn about the second stage of penetration testing, and this is the most important step since you will learn more about how to scan the servers and see what can be done to identify any vulnerabilities. In this stage, you need to take input from the discoveries made during the reconnaissance stage and gather information about the operating system and the users who use the information on that system. When you perform the testing as a system administrator, you can move back to the reconnaissance stage to see if there is any new information you need to obtain about the processes and people.

This stage's objective is for you to scan the information in the network and server and identify the information about the target. You can learn more about your information and network systems. During this stage, you need to focus on obtaining information about live hosts, device types(laptop, desktop, router, mobile, etc.), operating systems, software, public-facing services offered(SMTP, FTP, web applications, etc.). You can use this stage to remove any basic vulnerabilities in the system. These vulnerabilities are easy for one to detect during the scanning stage. You can use different tools in the scanning process, but we will look at the main tool Nmap that is used to perform this process. This stage's objective is for you to gather the information you can pass onto the next step of the ethical hacking process.

Network Traffic

You need to learn more about network traffic and how it works for you to understand the different tools and processes used in this stage. The communication which takes place between two devices over a network can only happen because of network traffic. The most popular modes of network traffic are wireless ethernet and wired ethernet. In this section, we will look at the main aspects of network traffic: firewalls and ports, and see how you can use them to improve your systems' security.

Firewalls and Ports

One of the main reasons why organizations have firewalls is to protect the systems, network, and server used to transfer information. You can protect the internal server and network on your system and server to ensure the communication between external and internal networks is secure. A firewall can is software or hardware which has rules to serve as a gatekeeper to a network. The firewall has access controls defined in them, enabling you to monitor the inbound traffic called outbound and ingress traffic. This is termed egress. The traffic that satisfies these rules ensures the firewall is never dropped or discarded during a hack. This is done by opening and closing ports on the firewall that allow or reject traffic.

We have looked at what ports are and how they are used and adjusted in the Linux operating system earlier in the book.

PING

Ping is an ICMP-based application to which you and every other user in the system or server is exposed to. When you use the ping command, you can send a code 0 and type 8 packet, indicating that the packet was just an echo. Systems that receive this package will instantly respond with a type 0 code 0 packet, an echo reply. A successful ping indicates to the system that the ping was made only on a live network. This means that you are working out of a live host. When you use the ping command, you can use the Windows command prompt. The ping will send the request at least four times by default. If you use this ping in the Linux terminal, the command will continue to run until you interrupt it.

Let us look at a successful and unsuccessful ping command.

If the host being pinged is Live

```
Ping 192.168.1.1
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=2ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time=1ms TTL=64
Reply from 192.168.1.1: bytes=32 time<1ms TTL=64
```

If the host is unreachable

```
Ping 192.168.1.200
```

Pinging 192.168.1.200 with 32 bytes of data:
Reply from 192.168.1.129: Destination host unreachable.
Reply from 192.168.1.129: Destination host unreachable.
Reply from 192.168.1.129: Destination host unreachable.
Reply from 192.168.1.129: Destination host unreachable.
Ping statistics for 192.168.1.200:
Packets: Sent 5 4, Received 5 4, Lost 5 0 (0% loss)

Traceroute

If you use the traceroute command, you will see that the ICMP ping command is used to determine the number of devices that lie between the system and the network. This is one way for it to initiate the trace on the target system. The traceroute command functions by manipulating the Time To Live value of a network packet, also known as TTL. This term indicates the number of times a packet of data is moved through a network and is broadcasted repeatedly by the host before the packet of data expires.

On Linux-based systems, the command to run a traceroute is traceroute. If you want to run the command on Windows, you can do it to check the vulnerabilities in the system and network. Let us take an example of a traceroute to google.com

tracert www.google.com

Tracing route to www.google.com [74.125.227.179]

over a maximum of 30 hops:

1 1 ms<1 ms 1 ms 192.168.1.1
2 7 ms 6 ms 6 ms 10.10.1.2
3 7 ms 8 ms 7 ms 10.10.1.45
4 9 ms 8 ms 8 ms 10.10.25.45
5 9 ms 10 ms 9 ms 10.10.85.99
6 11 ms 51 ms 10 ms 10.10.64.2
7 11 ms 10 ms 10 ms 10.10.5.88
8 11 ms 10 ms 11 ms 216.239.46.248
9 12 ms 12 ms 12 ms 72.14.236.98
10 18 ms 18 ms 18 ms 66.249.95.231
11 25 ms 24 ms 24 ms 216.239.48.4
12 48 ms 46 ms 46 ms 72.14.237.213
13 50 ms 50 ms 50 ms 72.14.237.214
14 48 ms 48 ms 48 ms 64.233.174.137
15 47 ms 47 ms 46 ms dfw06s32-in-f19.1e100.net [74.125.227.179]

Trace complete.

You can use different tools on Kali Linux to run these traces, and these tools

use the ICMP, TCP, and UDP protocols to scan the target systems and networks. The result of a successful scan will give you information such as operating systems, network hostnames, IP addresses, and services operated on the network. Some scanning tools can also be used to discover any vulnerabilities in the server and network. The details gathered in the scanning stage can be used in the exploitation stage to attack specific targets.

Nmap: The King of Scanners

The Nmap scanning tool is one of the most popular scanning tools used by system administrators since it is the easiest way to help you list down all the active hosts present on the target network. You can also use it to determine operating systems, services, ports, and even user credentials in some cases. This tool uses a combination of options, switches, and commands to find the details and vulnerabilities in your system during the scanning stage of this lifecycle.

Timing Options

We have learned above that the default timing option in the Nmap scan does not specify anything, but you can use the default value or choose from one of the options below. There is a feature in Nmap through which the user can specify the timing option he wants to use. This option can be used to override the default time option used in the tool. You can either choose to expedite the process or slow the process down if you need to.

Using this timing template, you can enable various settings, especially the one where you add a delay between the scanning and parallel processing of the ports. The different timing templates can be explained using the options scan_delay, max_scan_delay, max_parallelism. We can use these options to measure every timing template so that an appropriate timing template is used for scanning a target network.

You can use the can_delay option if you need to set the probes in the target system to a minimum number. You can use the max-scan_delay to define the scanner's maximum time between delays made while scanning the ports and IP addresses in the server and network. If you ask why this is important, it is because certain systems respond to probes only at a specific rate. When you use these options, you will adjust to the different probes and find a way to meet the system's requirements that you are scanning. The max_parallelism

option allows Nmap to send scan probes one at a time or in serial or in parallel. Let us look at the different timing options you can use in the Nmap tool.

-T0 Paranoid

When you use this timing option, the scan is performed slowly, and the network links are slow. You may get detected when you run these scans on your Linux network or server. Using this timing option, the scan will work serially and pause every 5 minutes. The max_delay setting is ignored during the scan when you use this timing template since the base scan_delay has a higher value than the default.

-T1 Sneaky

This timing method is used to perform a sneaky scan of the network and server. This is better and slightly faster than the previous timing by maintaining your discretion. This scan also serially scans the target system but reduces the scan_delay to about 15 seconds. The scan_delay is reduced in number, but since the value is higher, Nmap will ignore max_scan_delay.

Target

The target used in an IP address is an important part of the Nmap command string used, and if you end up using an incorrect target, you may end up scanning empty IP systems or spaces. You cannot do this as per the regulations. There are several ways to set a target for the Nmap scan, and we will look at the two common methods used to do this: IP address range and scan list.

IP Address Range

Using IP Address ranges to define targets in the system is an easy process to define the range or port you want to use. Let us take an example of a class C IP address range for our example. If you use the C IP address, you can run the scan on 254 different hosts. To do this, you can use the command below. Using this, you can scan all the hosts on a particular IP address range belonging to class C.

```
nmap 10.0.2.1 -255
```

You can use the Classless inter-domain routing (CIDR) to run the same scan

on the server or system's ports. CIDR is a quick way to specify an IP address range, but this is something beyond this book's scope.

```
nmap 10.0.2.1/24
```

If the IP address being used is very small, you can use a smaller range to define the target used in the scan. For instance, if you want to scan the IP addresses within the range, you can use the following command: nmap 10.0.2.1 -50

Scan List

You can use text files as inputs when you need to use the Nmap command to scan the list. The file will include the IP addresses which are present in your target server or system. You can choose to use the following IP addresses and store them in the target text file if you need to.

```
10.0.2.1
10.0.2.15
10.0.2.55
10.0.2.100
```

The syntax of the above command is as below:

```
nmap -iL target.txt
```

Port Selection

The Nmap command structure allows you to determine the ports you want to use by using the -p switch. This will help you determine the port you want to scan. You can either specify a single port or a range of ports depending on what you need to scan.

```
nmap -sS -p 1-100
nmap -sU -p 53, 25, 143, 80
```

You can also choose to combine both commands like the example below: nmap -sS -p 1-100, 53, 25, 143, 80

Output Options

The results of a scan you perform on the network and servers will scan the Nmap command results. You can print the information on the terminal window, but it is not always a good idea to do this. As the administrator, you need to save the file down with the output, so you can use it to determine how to remove any vulnerabilities in the system. You can redirect the output from

the scan and send it to a file. There are some built-in commands you can use if you prefer, and this allows you to redirect the output and move it onto a file. Let us go with each of these options.

-oN Normal Output

You can use this output option to create a normal text file to ensure the output is stored in the form of a text file. You can use this file to evaluate the output. You can also use it as an input key for other programs: nmap -oN output.txt 10.0.2.111

-oX Extensible Markup Language(XML) Output

Some applications and tools use XML files as their input when they perform a penetration test, and therefore, this option is extremely easy to use. You can also store the output in an XML format using the following command: nmap -oX output.txt 10.0.2.111

-oS Script Kiddie Output

The output files from a script kiddie attack cannot be used when you perform penetration testing, but these are a good way to determine how good the security in your network and server is. You cannot use the output from the following syntax for any industrial use.

```
nmap -oS output.txt 10.0.2.111
```

Nmap Scripting Engine

You need to learn how to build a custom script in Nmap if you do not want to use the built-in packages. If you are willing to use the preconfigured penetration tests in Nmap, you can use the following URL: https://nmap.org/nsedoc/.

For instance, you can use this script to fetch the MAC address and the NetBIOS, each of which is important information you need for the target system. You can use the flag '—script' as per the below example, along with the Nmap command followed by the script name.

```
nmap --script nbstat.nse 10.0.2.111
```

As a system administrator, you must ensure you have a script database and update it regularly. It is important for you to update the Nmap script as well

every time you perform a new penetration test on your server or network. To do this, use the following command: nmap --script –updatedb.

Conclusion

If you are a system administrator or engineer trying to learn more about Linux, then you found the right book. This book included all of the information you needed about Linux and how you can use it on your system or a virtual machine. You learned about the different distribution systems of Linux and how every distro differs from the other.

This book covered the different ways to encrypt and decrypt information. It also provided information about the different methods you can use to secure the server and the files and folders in the network and system. You learned about the different ways to use an access control list to manage how users access the system. It is important to take care of the access users have to the server and network since these individuals play a key role in taking care of the system's information. You learned more about the different ways to manage access control and discovered how you can manage privileges in the system. By the end of the book, you were able to learn to improve your network security, including the different methods you can use to test the system, and how to look for vulnerabilities.

Now it’s time to dive in and keep your system up to date and safe!

References

7 Tools to Encrypt/Decrypt and Password Protect Files in Linux. (2015). Tecmint.com. https://www.tecmint.com/linux-password-protect-files-with-encryption/

16 Ways to Secure a Linux Server. (2019, March 26). Liquid Web. https://www.liquidweb.com/kb/security-for-your-linux-server/

Access Control Lists(ACL) in Linux - GeeksforGeeks. (2018, May 2). GeeksforGeeks. https://www.geeksforgeeks.org/access-control-listsacl-linux/

comments, 08 O. 2019 P. H. M. F. 92up 6. (n.d.). *7 steps to securing your Linux server*. Opensource.com. https://opensource.com/article/19/10/linux-server-security

Controlling Unix & Linux Account Privileges: 9 Best Practices | BeyondTrust. (n.d.). Www.beyondtrust.com. Retrieved from https://www.beyondtrust.com/blog/entry/controlling-unix-linux-account-privileges-9-best-practices

Hoffman, C. (n.d.). *Why You Shouldn't Log Into Your Linux System As Root*. How-to Geek. Retrieved from https://www.howtogeek.com/124950/htg-explains-why-you-shouldnt-log-into-your-linux-system-as-root/

How to manage Linux user account security - Tutorial. (n.d.). UpCloud. Retrieved from https://upcloud.com/community/tutorials/manage-linux-user-account-security/

How to Setup Encrypted Filesystems and Swap Space Using "Cryptsetup" Tool in Linux - Part 3. (n.d.). Www.tecmint.com. Retrieved from https://www.tecmint.com/disk-encryption-in-linux/

S, R. (n.d.). *Encryption Methods in Linux | Unixmen*. Retrieved from https://www.unixmen.com/encryption-methods-linux/

skenlon. (n.d.). *Secure your Linux network with firewall-cmd*. Enable Sysadmin. Retrieved from https://www.redhat.com/sysadmin/secure-linux-network-firewall-cmd

The Best Linux Operating Distros. (2019, February 3). MUO. https://www.makeuseof.com/tag/best-linux-distributions/