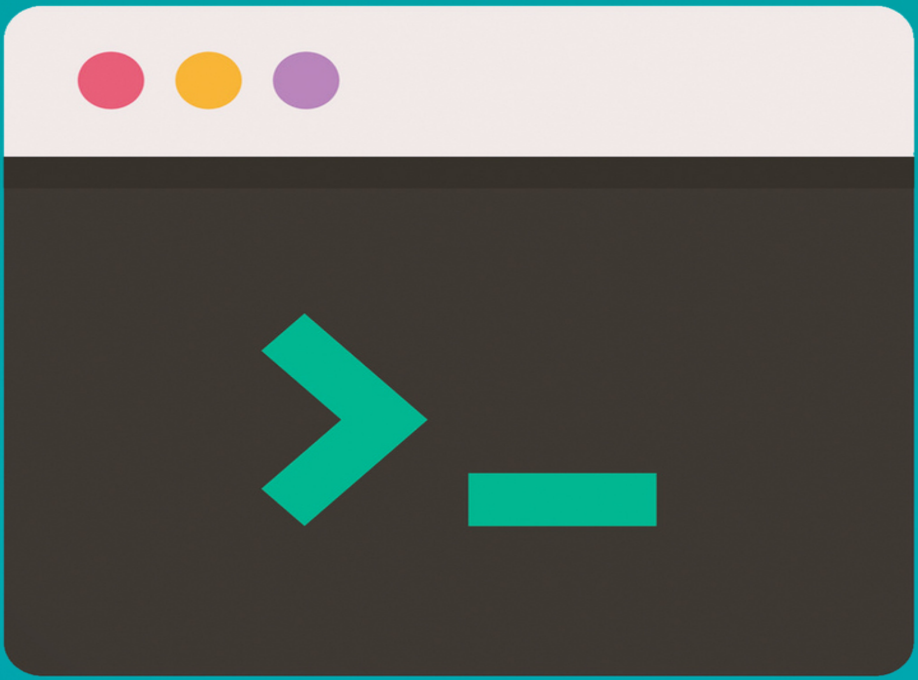
LINUX COMMAND LINE MADE EASY

A Practical, Step By Step Guide To Linux Commands For Beginners And Intermediates



CRAIG BERG

LIN

COM

LINE MA

A Practical, Step By Step

Linux Command Line Made Easy

A Practical, Step By Step Guide To Linux Commands For Beginners And Intermediates

Craig Berg

Introduction

Welcome to the Linux Command Line Guide For Beginners, and thank you for choosing this book as your pass to the Linux world.

Please note that this book is for beginners at Linux, not power users.

Whether you are coming from a version of Windows or OSx or giving Linux a try, this book will be invaluable to you.

However, while this book is indeed very beginner-friendly, it will not take you from a novice to a Linux command Line master in one night.

That's because the Linux Command Line is vast and challenging to master, especially for beginners. Learning it requires a great deal of determination and effort not because the Linux command line is complex or requires genius-level IQ, but because it's large and contains hundreds if not thousands of commands.

In all honesty, we can dedicate a single chapter to one command and all its arguments.

Nevertheless, the command line is a fundamental tool for Linux users. The command line brings you every possible tool to work with in the ins and outs of your operating system with just a keyboard. If you are skeptical about this, follow the book, and you will realize real power.

The book has an efficient organizational structure:

1. The first chapter gives you the basics and must-know elements of working with the Linux Command line.
2. In the second chapter, we shall look at Linux configuration and where config files of applications are stored, and how to manage them.
3. In the third chapter, we shall dive into the Linux tools and common commands you will find yourself using most

of the time as you work with the command line.

4. Finally, we will conclude with a discussion on Linux permissions, groups, and more.

In simple terms, Linux and Linux command line is amazing because it opens many opportunities for you.

In this beginners' book, we do not assume you have any programming knowledge. That's why the book will walk you through every step.

Read the book from start to finish because its structure is a lot like a syllabus and less like a quick reference guide.

All you need is a computer and internet access for Linux installation —more on that later. Once you have what you need, we can get started with the installation and hands-on typing command lines, the fun part!

PS: I'd like your feedback. If you are happy with this book, please leave a review on Amazon.

Please leave a review for this book on Amazon by visiting the page below:

https://amzn.to/2VMR5qr

Table of Content

Section 1
Linux Installation Guide

The first thing we will do before diving into the command line is getting your system setup. For Windows or PC users, you have three available options that allow you to run Linux on any system running Microsoft office. We will cover each option individually in the upcoming sections.

These available options for PC users include:

- Using Linux as Dual Boot with Windows,
- Installing Linux on a Virtualization software,
- Windows subsystem for Linux (WSL) features and finally
- Using Live Media such as USB or DVDs.

For Mac OS users, you can use Dual Boot alongside Linux, use Virtualization software, or opt for live media.

Let us get started.

How to Choose A Linux Distribution

This section covers how to choose the right Linux distribution.

A Linux distribution, commonly called a distro, is a different “flavor” of a Linux operating system built on top of the native Linux kernel to carry out various use-cases and preferences.

Since all distributions are Linux-based, they tend to be quite similar, and therefore usable interchangeably. For example, a distribution like Ubuntu is the most popular due to its ease of use and the ability to abstract small configuration tasks for its users. A distribution such as Arch Linux does not offer the same features and is better suited if you want to have more control over the system and tweak it to suit your needs.

If you are entirely new to Linux, opt for a Debian distro such as Ubuntu or Linux Mint, Arch distros such as Manjaro are also supported by this book.

Please do not take this recommendation to mean that these distributions are beginner-geared. This book recommends these distros because they provide a familiar environment for new users.

Installing Linux on Windows: Virtualization.

The first method we will explore for PC users is using Virtualization Software. This is the easiest method as it does not require you to change any system configuration or settings. All you require is a piece of simple-to-use software.

The first step is to download the iso file for your Linux distribution. For this book, we will opt for Ubuntu 20.04.1—the current version as of writing this book.

Open the browser, navigate to the resource link below, and download the Ubuntu iso.

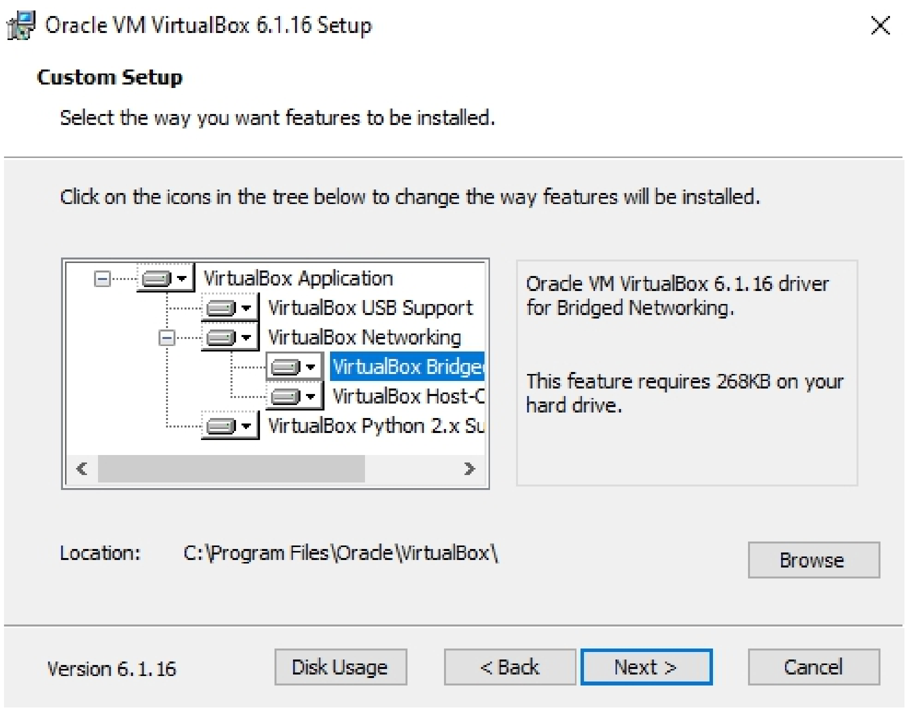
https://ubuntu.com/

Next, we need to get virtualization software. We are going to use Oracle VirtualBox as it is easy to install and use. You can also choose other tools such as VMware, KVM, or Hyper-V. Open your browser, navigate to the resource link below, and download the latest version of VirtualBox.

https://www.virtualbox.org/

NOTE: You will have to enable virtualization technology (VT-X) to run any virtualization software. Check the web on how to enable Virtualization for your specific device.

Once you download the VirtualBox installer, launch it and go through the typical Windows installation process. Ensure to select all the features during the installation process as shown below.


Oracle VM VirtualBox 6.1.16 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridge
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

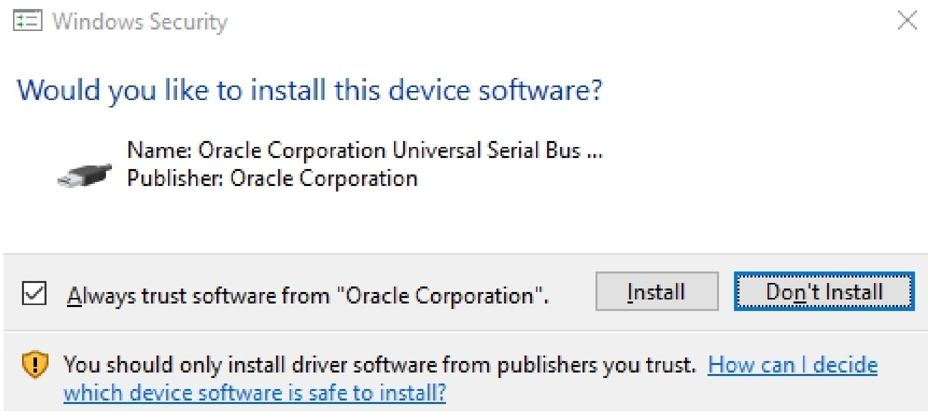
Oracle VM VirtualBox 6.1.16 driver for Bridged Networking.

This feature requires 268KB on your hard drive.

Location: C:\Program Files\Oracle\VirtualBox\

Version 6.1.16

During the installation process, the setup may prompt you to install the Oracle USB driver. Select **install.**


Windows Security

Would you like to install this device software?

Name: Oracle Corporation Universal Serial Bus ...
Publisher: Oracle Corporation

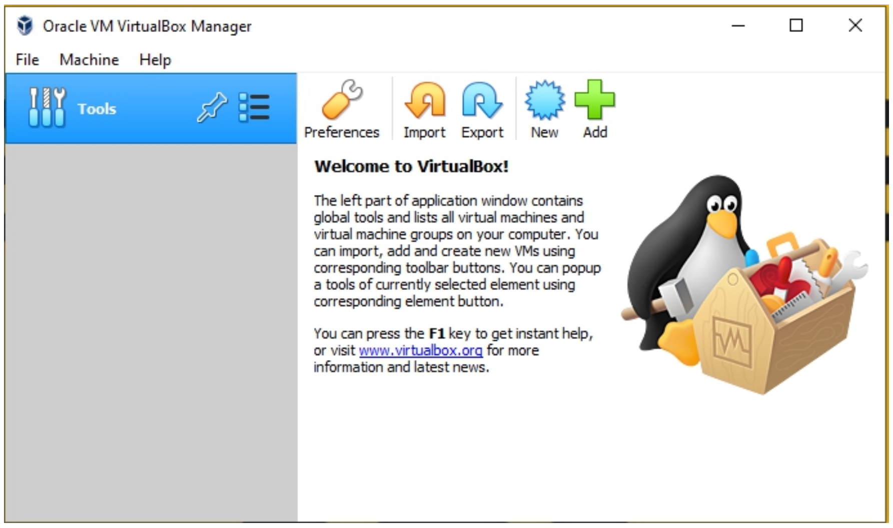
☑ Always trust software from "Oracle Corporation".

You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

With the software setup, we can begin installing Ubuntu or your Linux distribution.

In most cases, the distribution you choose will provide an installation guide. Check the guide for your specific distribution.

Once you launch VirtualBox, you will get a window similar to the one shown below:



Welcome to VirtualBox!

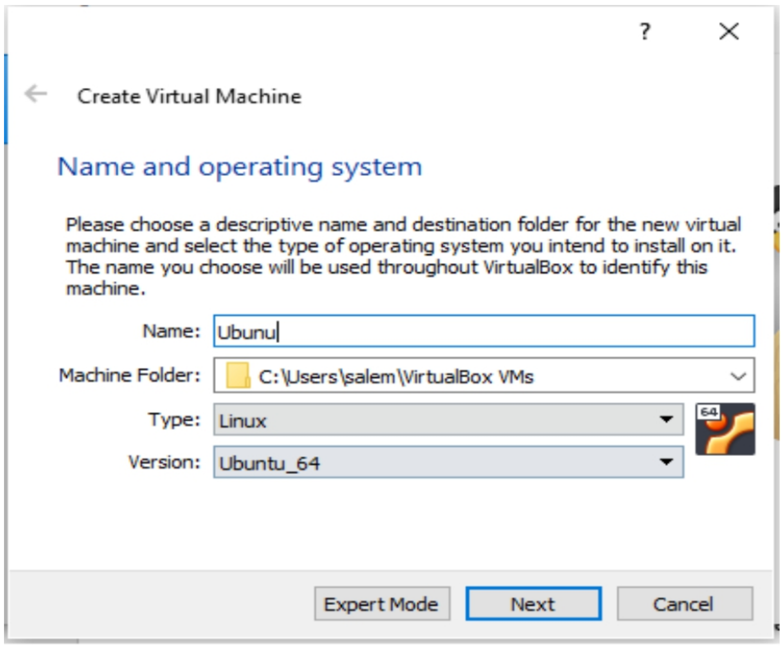
The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Next, click on New, which will launch a box that requires you to provide basic details about your Operating system. Enter the name of the VirtualBox—feel free to provide any suitable name. In this case, we shall name it Ubuntu.

Next, select the location in your filesystem where you want the Virtual machine file stored. You can leave this as default. Next, choose the type from the dropdown menu. If you provide a name for popular Operating Systems, VirtualBox should detect the Type and Version. If this does not happen, select the type as Linux and type as Ubuntu_64.

Select Debian for all Debian derivatives and Arch for Arch-based distros.


Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: Ubunu

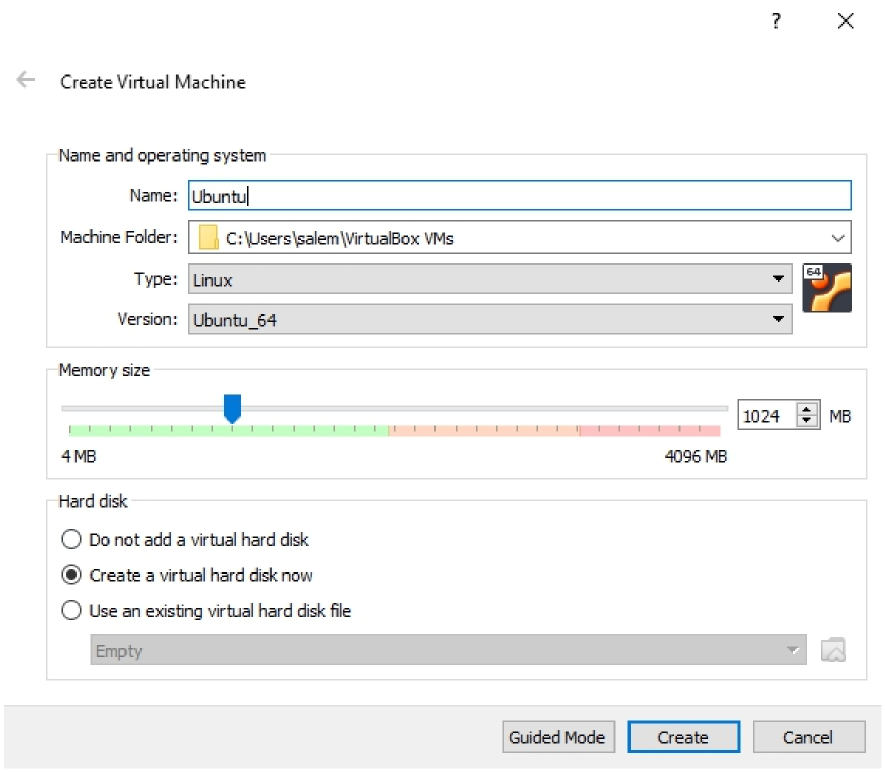
Machine Folder: C:\Users\salem\VirtualBox VMs

Type: Linux

Version: Ubuntu_64

Next, select Expert mode, which will provide you with more options. Select the amount of memory you want to allocate for your OS. We recommend above 4GB unless you have less, in which case, you can allocate 1- 2GB.

Next, select create virtual disk now and click on create now:


Create Virtual Machine

Name and operating system

Name: Ubuntu

Machine Folder: C:\Users\salem\VirtualBox VMs

Type: Linux

Version: Ubuntu_64

Memory size

1024 MB

Hard disk

○ Do not add a virtual hard disk

● Create a virtual hard disk now

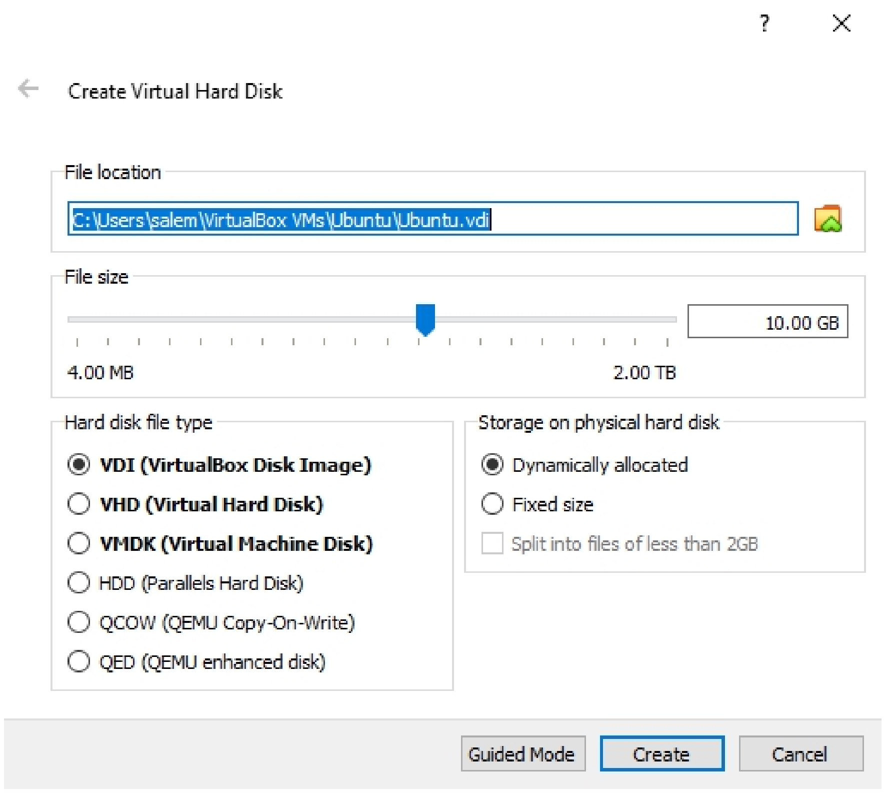
○ Use an existing virtual hard disk file

Empty

Select the size and location storage for the virtual disk in the next windows—leave this as default.

In hard disk file type, select VirtualBox Disk image and set as dynamically allocated. That ensures the size of the hard disk increases as the system needs more storage.

Click on create.



Create Virtual Hard Disk

File location: C:\Users\salem\VirtualBox VMs\Ubuntu\Ubuntu.vdi

File size: 10.00 GB (4.00 MB – 2.00 TB)

Hard disk file type:
- [✓ VDI (VirtualBox Disk Image)]
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)
- HDD (Parallels Hard Disk)
- QCOW (QEMU Copy-On-Write)
- QED (QEMU enhanced disk)

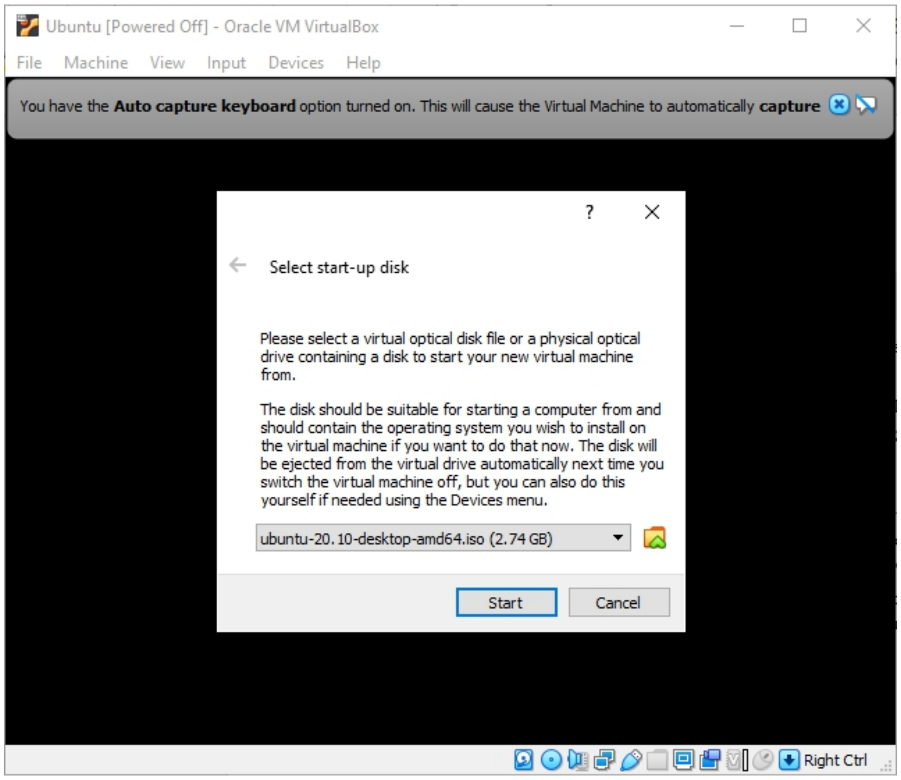
Storage on physical hard disk:
- [✓ Dynamically allocated]
- Fixed size
- Split into files of less than 2GB

That should create a virtual machine that should be available in the list of your virtual machines.

Now click on the Settings icon to launch the virtual machine's settings, then next, select the System option on the left side menu. That will give you all the settings on the virtual system of your virtual machine. Under processor, select the number of processors to allocate the machine.

Next, go to storage settings and select the Controller IDE. Under attributes, click on the disk icon and select "Choose a disk file." This will launch the file explorer. Navigate to your download folder and select the Ubuntu iso image we downloaded earlier.

Click on save and go back to the main menu. Select the start option to boot up the virtual machine. During bootup, VirtualBox will prompt you to select the startup disk. Select the Ubuntu iso we configured.


Ubuntu [Powered Off] - Oracle VM VirtualBox
File Machine View Input Devices Help
Start
Cancel
Right Ctrl

You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture**

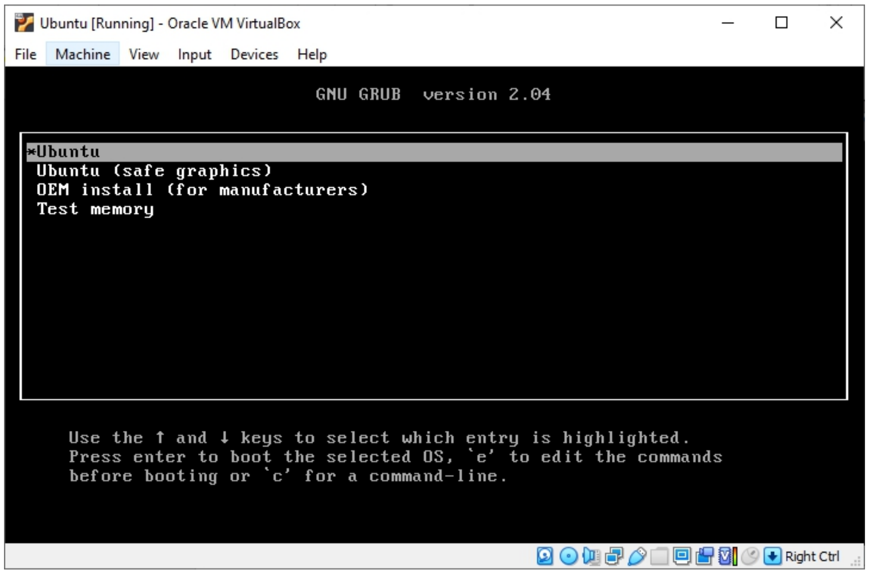
Select start-up disk

Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

ubuntu-20.10-desktop-amd64.iso (2.74 GB)

That will boot you up in the GRUB menu to select which option to use. Select Ubuntu in the first option—as shown below.


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
GNU GRUB  version 2.04

*Ubuntu
 Ubuntu (safe graphics)
 OEM install (for manufacturers)
 Test memory

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line.
```

If all configurations and settings are working properly, this should boot up using the Ubuntu installer from the iso. Once in the installer menu, select Install Ubuntu as we are running it on a virtual machine.

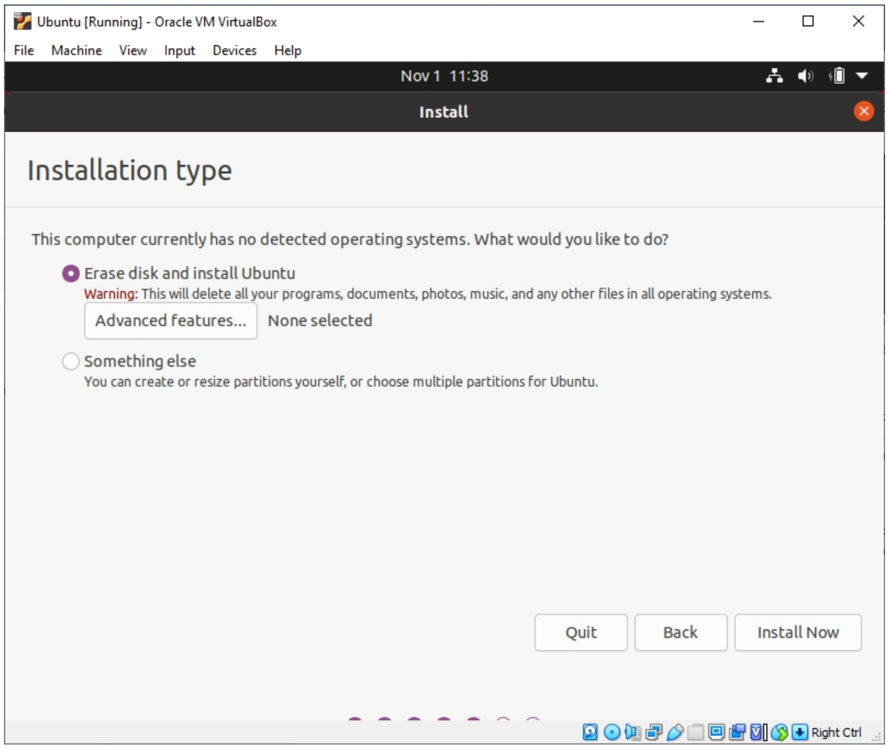
Continue with the installation options such as Language and Keyboard Layout. Next, you can select either Normal installation, which comes prepackaged with games and other utilities, or Minimal install, which only contains basic system utilities.

Since we will work with the terminal, it's best to choose the Minimal install option.

You can enable or disable to install updates during the installation.

In the next section, select erase disk, and install Ubuntu. If you want to customize disk allocation, choose advanced options.

As we are running a virtual machine that only affects the virtual disks, we can use the entire disk for the installation.



Installation type

This computer currently has no detected operating systems. What would you like to do?

Erase disk and install Ubuntu

Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

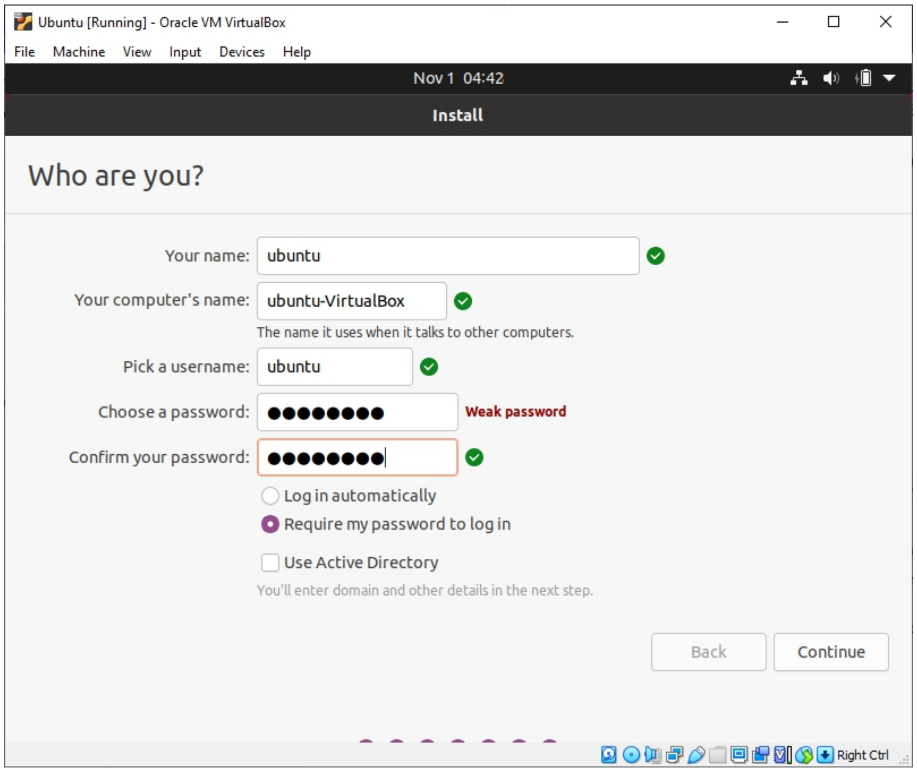
Advanced features... None selected

Something else

You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Select install now, which will prompt you to confirm changes to the disk. Go ahead and click yes and confirm.

Confirm your location and click on next. In the next step, enter your details and set up your username and password, and click on continue.


Ubuntu [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Nov 1 04:42
Install

Who are you?

Your name: ubuntu

Your computer's name: ubuntu-VirtualBox

The name it uses when it talks to other computers.

Pick a username: ubuntu

Choose a password: ●●●●●●●● Weak password

Confirm your password: ●●●●●●●●

Log in automatically [✓ Require my password to log in]

Use Active Directory

You'll enter domain and other details in the next step.

Finally, click "install" to begin the installation process. Once the installation is complete, you should boot into Ubuntu and start working with Linux.

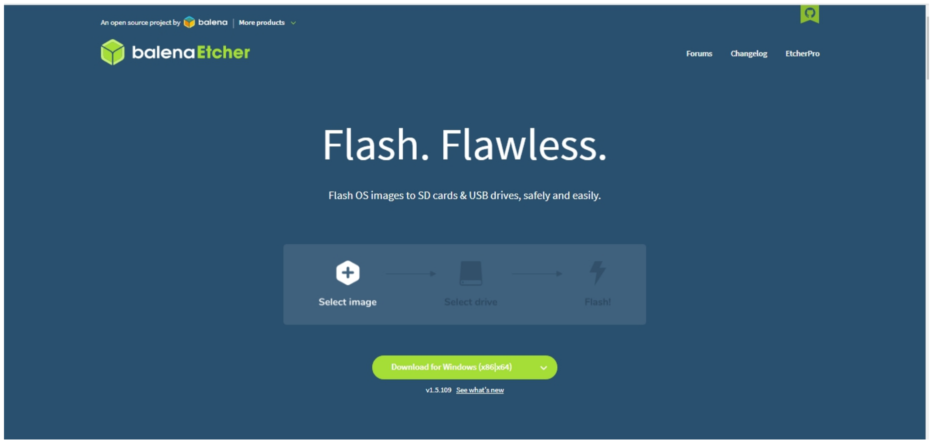
Installing Linux on Windows: Dual Boot

Here, we are going to cover how to dual boot Windows alongside Ubuntu. For this method, you will require a USB flash drive and free space on your hard disk.

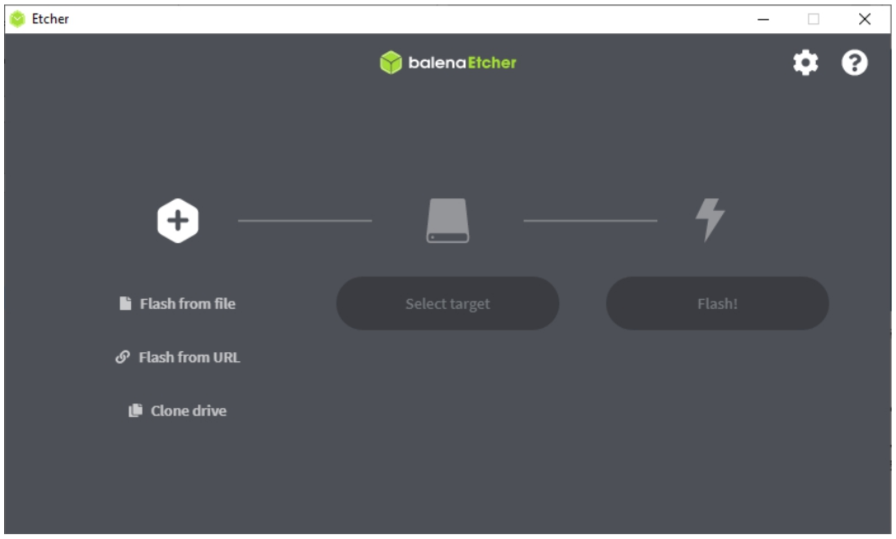
The first step—as usual—is to download the Ubuntu iso from the official site. Next, we are going to download a tool to burn the iso on the flash drive.

Open your browser, navigate to the resource page below, and download the windows installer.

https://balena.io/etcher


An open source project by balena | More products
balenaEtcher
Forums
Changelog
EtcherPro
Flash. Flawless.
Flash OS images to SD cards & USB drives, safely and easily.
Select image
Select drive
Flash!
Download for Windows (x86|x64)
v1.5.109 See what's new

Once you have acquired the installer, install and launch to begin the process. Once you launch Balena, you will get a window that resembles the one shown below:



In the window above, select the “Flash from File” option. Doing this will prompt you to provide the location of the Ubuntu image we downloaded.

Next, Insert your Flash disk into your computer; BalenaEtcher should automatically detect it.

If you have more than one device connected, choose the correct from the Target menu. Once selected, click on Flash to begin the burning process.

NOTE: This will wipe your USB drive. Backup important data!

The next thing we are going to do is create a partition for our Linux installation. On your Windows machine, Right-click the start menu and launch the Disk management tool.

On the partition you want to create some space, right-click and select Shrink Volume. Provide the amount of space to allocate to the partition; the recommendation is more than 10GB of storage.

Once completed, reboot your machine and select your boot devices as the USB device we created above.

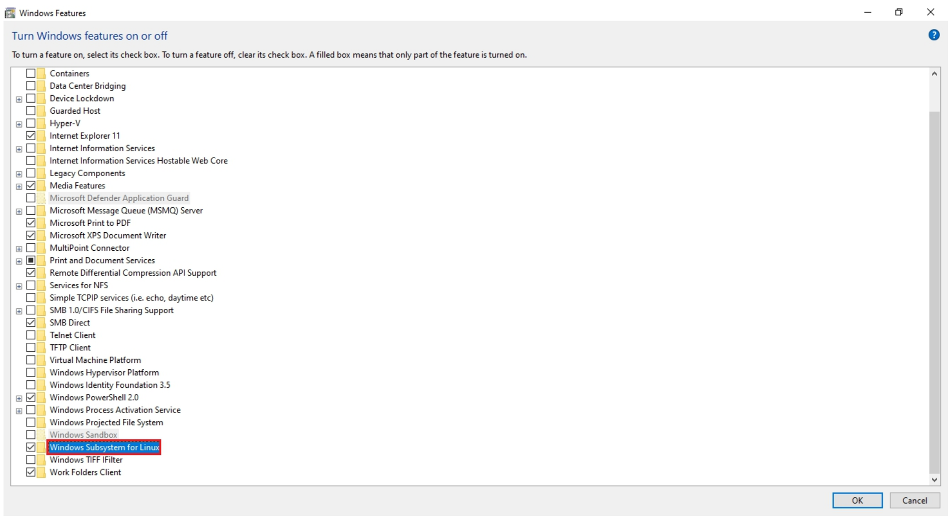
NOTE: Check the web on how to select a boot device for your specific machine.

This will boot Ubuntu and take you to the installation window. Follow the process in the above section until you get to a point where the system prompts you to select the installation type. Select install Alongside window and click continue and complete the process.

Installing Linux on Windows: WSL

In a recent version of Windows, Microsoft has introduced a feature that allows you to run Linux from within your system without a Hypervisor. However, this feature is only available in the most recent version of Windows 10. If you have earlier versions of Windows, consider using other methods to run Linux.

For this to work, we need to enable the WSL feature. Open the start menu and search for "Turn Windows features on or Off." This will launch the windows features window where we can turn on the WSL feature.

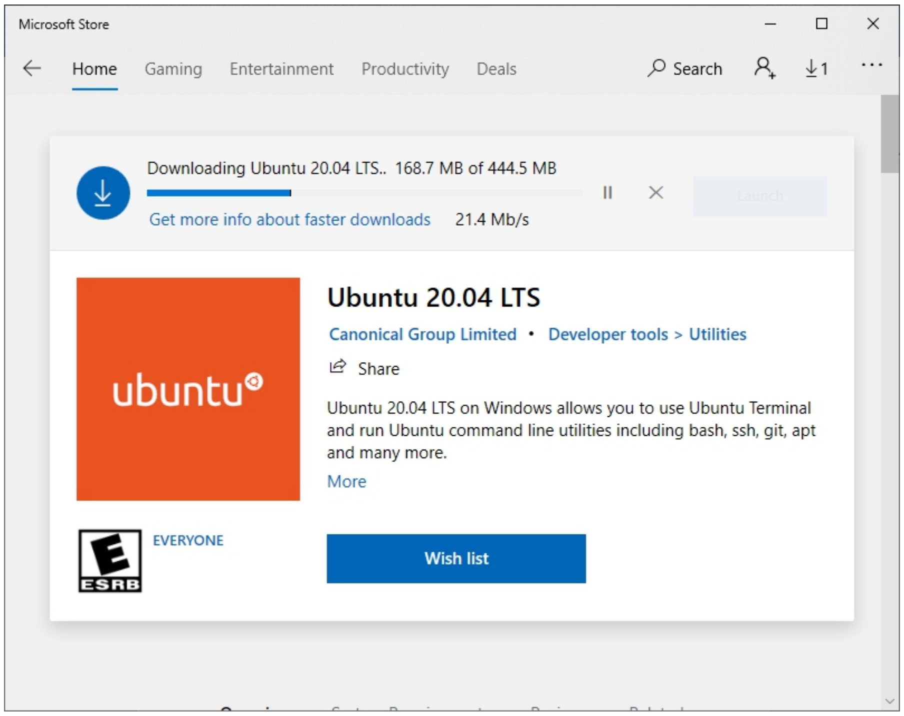

Windows Features

Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ Containers
- ☐ Data Center Bridging
- ☐ Device Lockdown
- ☐ Guarded Host
- ☐ Hyper-V
- ☑ Internet Explorer 11
- ☐ Internet Information Services
- ☐ Internet Information Services Hostable Web Core
- ☐ Legacy Components
- ☑ Media Features
- ☐ Microsoft Defender Application Guard
- ☐ Microsoft Message Queue (MSMQ) Server
- ☑ Microsoft Print to PDF
- ☑ Microsoft XPS Document Writer
- ☐ MultiPoint Connector
- ■ Print and Document Services
- ☑ Remote Differential Compression API Support
- ☐ Services for NFS
- ☐ Simple TCPIP services (i.e. echo, daytime etc)
- ☐ SMB 1.0/CIFS File Sharing Support
- ☑ SMB Direct
- ☐ Telnet Client
- ☐ TFTP Client
- ☐ Virtual Machine Platform
- ☐ Windows Hypervisor Platform
- ☐ Windows Identity Foundation 3.5
- ☑ Windows PowerShell 2.0
- ☐ Windows Process Activation Service
- ☐ Windows Projected File System
- ☐ Windows Sandbox
- ☑ Windows Subsystem for Linux
- ☐ Windows TIFF IFilter
- ☑ Work Folders Client

Click on Ok to apply the changes. This process may require you to reboot your machine. Next, open Microsoft store and search for Ubuntu. Select Ubuntu 20.04 LTS and click on install.


Microsoft Store

Downloading Ubuntu 20.04 LTS.. 168.7 MB of 444.5 MB

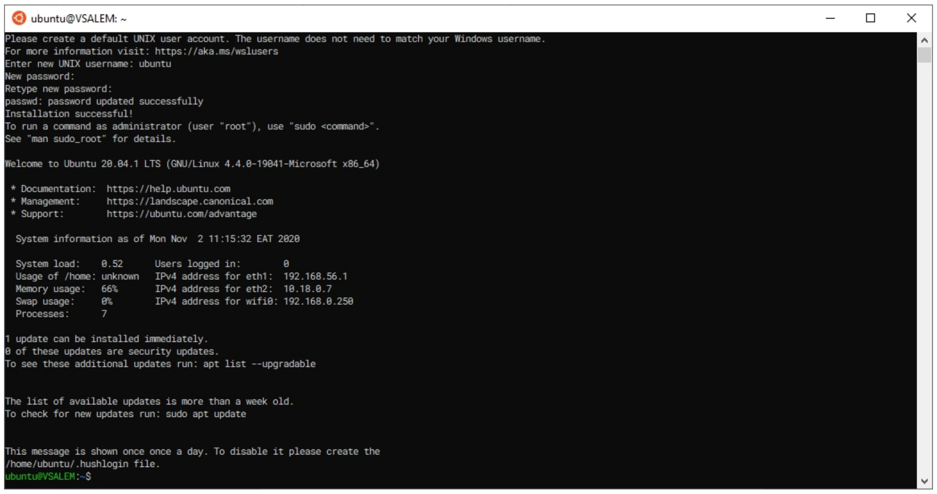
Get more info about faster downloads 21.4 Mb/s

Ubuntu 20.04 LTS

Canonical Group Limited • Developer tools > Utilities

Ubuntu 20.04 LTS on Windows allows you to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

Once the WSL feature is enabled and Ubuntu installed, Launch the application to start the installation. Provide a Linux username and password to boot up the machine.



ubuntu@VSALEM: ~

```
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: ubuntu
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Nov  2 11:15:32 EAT 2020

  System load:    0.52      Users logged in:       0
  Usage of /home: unknown   IPv4 address for eth1:  192.168.56.1
  Memory usage:   66%       IPv4 address for eth2:  10.18.0.7
  Swap usage:     0%        IPv4 address for wifi0: 192.168.0.250
  Processes:      7

1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


This message is shown once once a day. To disable it please create the
/home/ubuntu/.hushlogin file.
ubuntu@VSALEM:~$
```

NOTE: WSL Installation does not provide a GUI interface, but since this guide is about working with the terminal, this should not be a problem.

Installing Linux on Mac OSx: Virtual Machine

Because of this book's scope, we shall not discuss a step-by-step process to install Linux on a Virtual Machine on Mac. Check the Linux installation process for Windows as they are similar.

Section 2

Linux Command, Terminal, and Shell

Before we dive deep into executing Linux commands, let us first define a few common terms in Linux.

Although these terms are basic, it's good to get a good handle on their meaning and differences:

Linux Dictionary

The most common Linux terms you need to know are:

1. A command – A command in Linux refers to a set of characters recognized by the kernel and used to perform various tasks.
2. A Terminal – Terminal refers to a graphical interface used to take the user's command and bash it along to the shell. It usually includes a shell prompt with the format of username@machinenane and a cursor. Although you will find various terminal versions in different distros, they mainly carry out the same tasks.
3. Shell – The shell refers to the middleman between the user and the kernel. It allows you to enter commands and interprets them to the kernel. Linux includes various types of shells, such as bash, zhs, fish, dash, etc.

Since you now know the most important terms, let's get started with our first command in Linux.

If you are using a virtual machine or dual boot, open the Applications menu, and search for the terminal.

For WSL users, launch the Ubuntu application; you should drop into a shell immediately.

Once in the shell, you should see something like this:

```
ubuntu@VSALEM:~$
```

We call that a shell prompt; it indicates that the shell is ready to access commands and execute them. The shell prompt may vary depending on your distribution but mainly includes your username@machinename, followed by a dollar sign. If you have a pound sign # rather than a dollar $ sign, it means you are running the shell session with superuser privileges. That means you have logged-in in as a superuser or launched a terminal that includes root privileges.

If the terminal is ready and all things look good, let us try to enter some rubbish text and see what happens.

```
ubuntu@VSALEM:~$ dfsskdhd
```

Since the command entered does not make any sense to the shell, it shall give you an error and provide a prompt to enter another command.

```
ubuntu@VSALEM:~$ dfsskdhd

dfsskdhd: command not found

ubuntu@VSALEM:~$
```

The Command Line History

The shell gives us the ability to view all the commands we execute. We can view this by pressing the up or down arrow key. This feature is called The command Line History; it can store over 500 commands.

Simple Commands

Since we have verified that the terminal and shell are working, let us execute valid commands.

The first command we are going to type is pwd , used to print the current working directory.

```
ubuntu@VSALEM:~$ pwd

/home/ubuntu
```

The command tells us that we are in the /home/ubuntu directory—we'll learn more about directories in upcoming chapters.

Try another command such as free to see the total amount of free memory in your system.

```
ubuntu@VSALEM:~$ free
              total        used        free      shared buff/cache   available
Mem:        4127084     2684684     1213048       17720      229352
    1308668
Swap:      12582912      164640    12418272
```

Once done with our terminal session, we can close it by terminating the terminal window or using the shell prompt's exit command.

```
ubuntu@VSALEM:~$ exit
```

NOTE: Even when you are running a terminal session as a user, various terminal sessions are always running in the background. These terminal sessions are known as virtual consoles and are accessible by pressing CTRL + ALT + F1

Section 3
Navigating the Filesystem

In this chapter, we will learn how to navigate the Linux filesystem using the command line.

Navigating the filesystem using the command line is an essential skill as most of the time, you will be working with files.

Now, before jumping directly into the terminal, let us talk about Linux Filesystem.

Linux and all other Unix-like Operating systems use the hierarchical directory file structure. The hierarchical directory structure is where directories and files have a tree-like filesystem or organizational structure. Directories are also known as folders.

The first directory in the hierarchical structure is called the root directory. The root directory also contains subdirectories and other files within it. These subdirectories also contain other directories and files and so on.

You can learn more about HFS here:

https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

https://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/Linux-Filesystem-Hierarchy.html

For Windows-like systems, you will find each specific filesystem structure for each storage device.

In Linux, a single system contains a single filesystem structure despite the number of storage devices connected. Each storage device connected gets mounted on a certain point in the filesystem.

The Current Working Directory

Those used to working with the graphical system navigate the directory easily as it's just going up and down.

However, the terminal does not provide a graphical method to access files and directories, making navigating the Linux Filesystem not easy. It's a complex system, and without knowing where we are or where we want to go, we can't do much.

NOTE: At each point in the terminal, you are inside a specific directory. Some terminals may show you which directory you are in at that moment.

We can view which directory we are in by using the pwd command. Unless you navigate to a different location in the file system once you launch a terminal session, your current directory is your home directory. Each account in Linux is assigned their home directory where that specific account is allowed to read, write and execute files in that directory.

A root user or users in sudo group can access all files and directories within the filesystem.

Listing Files and Directories

To list files and directories within the current working directory, we use the ls command.

```
ubuntu@VSALEM:~$ ls
```

```
Desktop Documents Music Pictures Public Templates Videos
```

The ls command is an amazing command that you can use to do more than list files and directories of the current directory. We will discuss the ls command a bit more deeply in later sections.

Moving Between Directories

To move between directories. i.e., from our current directory within the filesystem to another location in the filesystem, we use the cd command. The cd command is followed by the *pathname* of the directory we want to navigate to.

A *pathname* refers to the directories or subdirectories we navigate to get to that directory.

If that sounds confusing, here’s a simpler explanation:

Consider an instance where we want to navigate to a directory called music. To get there, first, we go to home, then the username, then

Downloads, and then music. The *pathname* is thus /home/username/Downloads/Music.

Pathnames can be provided in two ways, i.e., Relative and Absolute pathways.

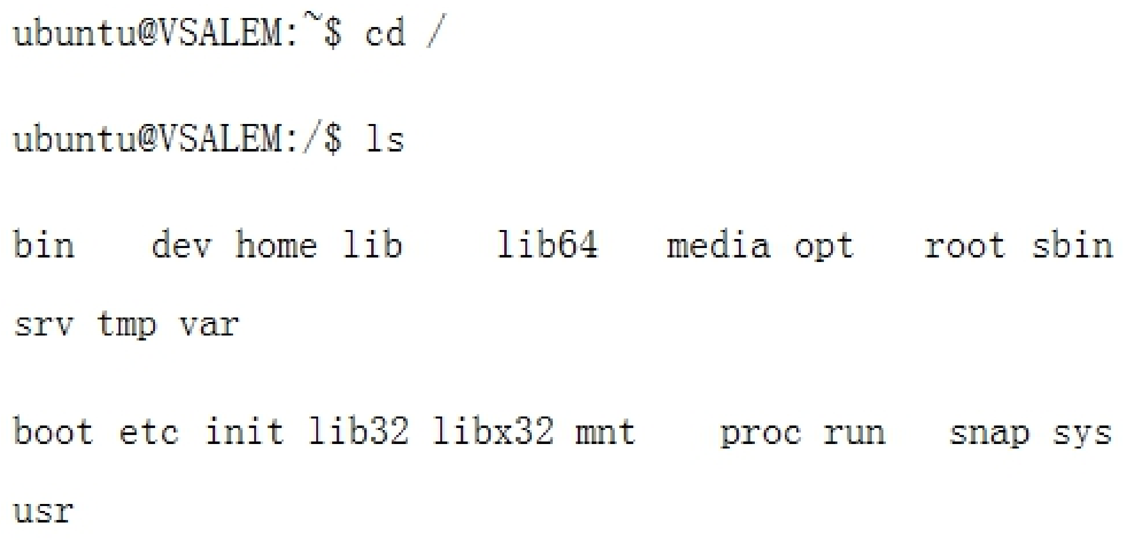
Let us discuss the difference between these pathname styles:

1: Absolute Pathnames

Absolute pathnames are very common when working with Linux directories. They mainly begin with the root directory (uppermost directory) and follow the tree (subdirectories) branch by branch to the target location.

An example of an absolute path is the one shown above.

Let us talk about where devices get mounted in Linux. The path to where devices get mounted by default is /media/username/<name of storage device>. That means that, from the root directory indicated by /, we go to the media directory, then username directory, and finally, the device name. If you want to see the top directory, cd into / to view all the files.



```
ubuntu@VSALEM:~$ cd /

ubuntu@VSALEM:/$ ls

bin   dev home lib    lib64   media opt   root sbin

srv tmp var

boot etc init lib32 libx32 mnt    proc run   snap sys

usr
```

Depending on the distribution you are using, you may get additional or fewer directories than those shown above.

You may notice that once we navigate to a specific directory, the shell prompt style changes appending the current path before the dollar sign.

For example, change directory to /usr/share using the command cd /usr/share

```
ubuntu@VSALEM:/$ cd /usr/share

ubuntu@VSALEM:/usr/share$
```

2: Relative Pathnames

Relative pathnames are the other type of pathnames in Linux. Unlike absolute pathnames where you start at the root directory, relative pathnames start at the current working directory.

To achieve this, relative pathnames use special symbols to indicate relative positions in the filesystem. These symbols include the dot (.) and the dot dot (..)

The dot symbol represents the current working directory while the dot dot symbol represents the working directory parent.

Let us see how it works.

In your terminal, navigate to /usr/share or any directory where you have read permission.

```
ubuntu@VSALEM:~$ cd /usr/share/

ubuntu@VSALEM:/usr/share$ pwd

/usr/share
```

Suppose we want to go to the parent directory of share, which is /usr? We can either use the absolute path and cd /usr or use the relative path by entering the command cd ..

Now the question becomes, which should you use? The answer is the one that requires less typing.

For example:

Let us assume you are in your /home folder. To go to /usr/share we can go using cd /usr/share or cd ../../usr/share

NOTE: If you do not include the . (representing the current directory), the shell automatically includes it

For example:

```
ubuntu@VSALEM:~$ cd /usr

ubuntu@VSALEM:/usr$ cd share

ubuntu@VSALEM:/usr/share$ cd ..

ubuntu@VSALEM:/usr$ cd ./share

ubuntu@VSALEM:/usr/share$
```

That means if the working directory is not specified, the shell automatically assumes.

Tips and Notes about Linux Filesystem

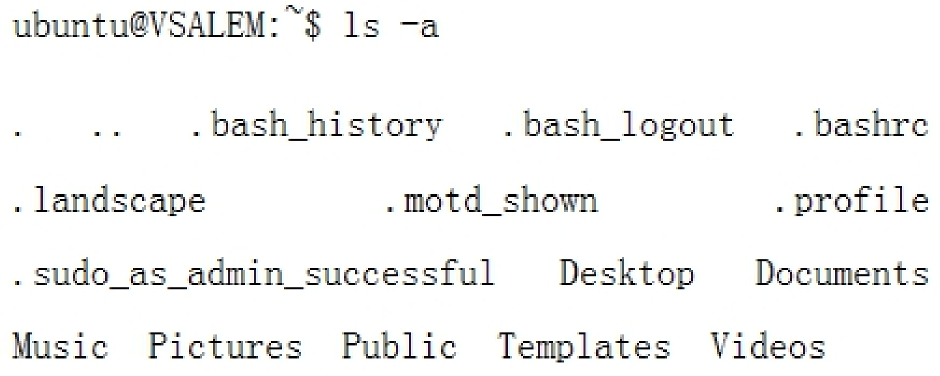
Here're some tips and pointers on working with the Linux filesystem:

1. Use the cd command to go back to your home directory in an instance.
2. cd - - The cd command with the dash (-) argument takes you back to the previous working directory.
3. cd ~ - Changes to the directory of the username specified.
4. Filenames in Linux are case sensitive. Thus, *Directory* and *directory* are two different files.
5. Avoid using spaces while naming files and directories. Although Linux does support spaces, use underscores, dots or dashes because navigating directories with spaces is very hard. Here's an example:

   ```
   ubuntu@VSALEM:~$ cd This\ directory\ has\ spaces/
   ```

   ```
   ubuntu@VSALEM:~$ cd This.Dir.Has.Spaces/
   ```

6. Linux contains hidden files that start with a dot (.). To view hidden files and directories, use ls -a, which includes the hidden attribute.



```
ubuntu@VSALEM:~$ ls -a

.    ..    .bash_history    .bash_logout    .bashrc

.landscape          .motd_shown          .profile

.sudo_as_admin_successful    Desktop    Documents

Music  Pictures  Public  Templates  Videos
```

Linux does not contain or recognize the concept of file extensions. However, that does not mean that if Linux does not, programs running in Linux don't.

ls Command Deep Dive

As you continue to work with Linux, you will realize that ls is probably one of the most used commands. That is perhaps for a good reason because it provides you with the ability to view files and directories, their attributes and permissions, and much more.

In this subsection, we shall delve deeper into the ls command and its arguments and learn how to use them.

Up to now, we have covered how to list files and directories within our current working directory.

Apart from the current directory, we can specify which directory to list without moving into it.

For example:

```
ubuntu@VSALEM:~$ ls /usr/share

GConf             consolefonts gettext            libc-

bin                  pastebin.d           sounds

PackageKit                     consoletrans   git-core

libdrm                    perl                systemd

X11                           cryptsetup        gitweb

lintian                   perl5               tabset

-----------------------truncated output---------------

----------
```

From the above command, we can see the contents of the /usr/share directory directly from the home directory.

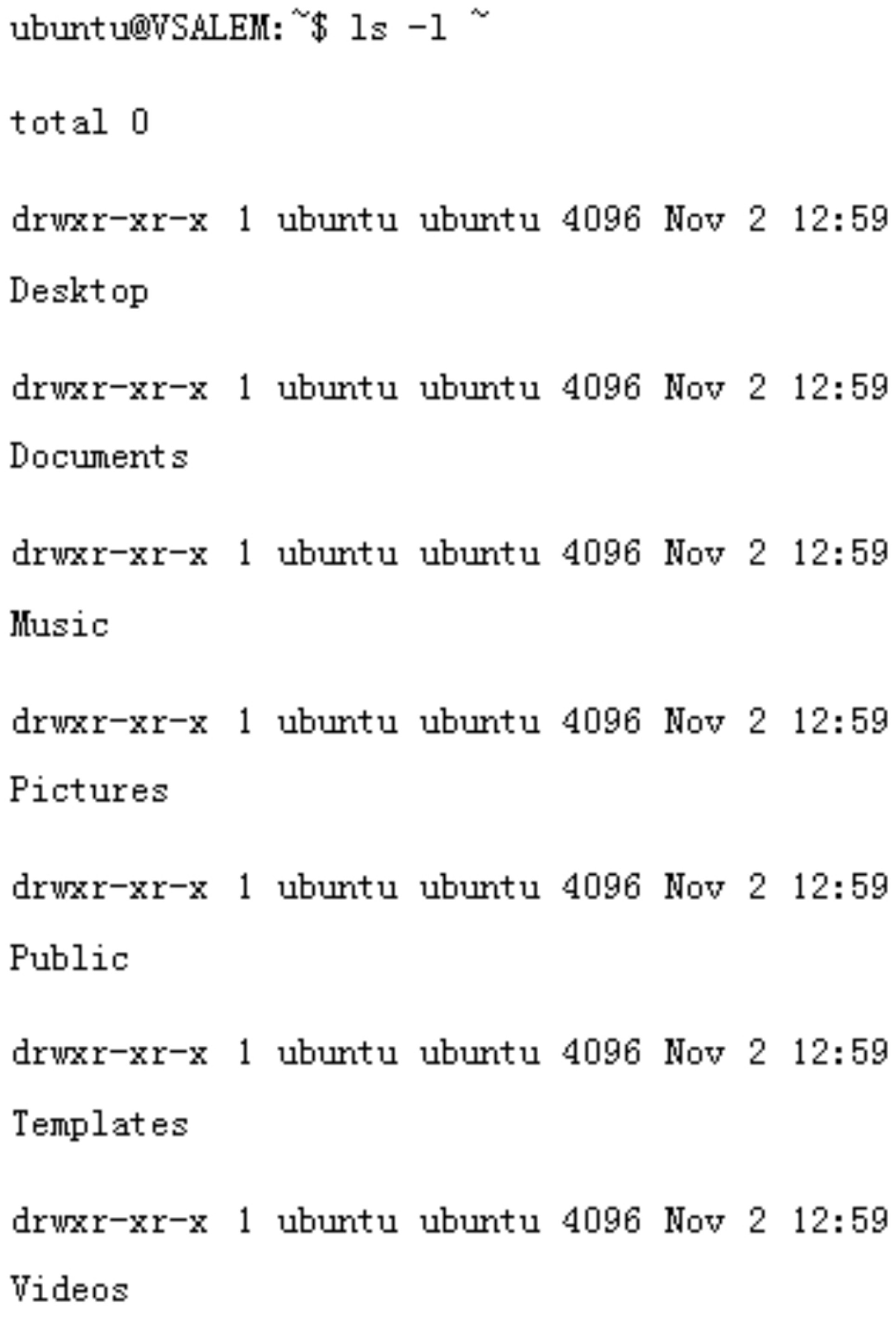
You can also list more than one directory using a single command, shown below:

```
ubuntu@VSALEM:~$ ls ~ /usr

/home/ubuntu:

Desktop  Documents  Music  Pictures  Public  Templates

Videos

/usr:

bin games include lib lib32 lib64 libexec libx32 local

sbin share src
```

The command above lists files and directories in both the home directory (~) and /usr directory.

We can also pass some more arguments to the ls command to get more details about the files and directories.

a. The -l argument changes the format which the files and directories are listed and provides more details. Example:



```
ubuntu@VSALEM:~$ ls -l ~

total 0

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Desktop

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Documents

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Music

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Pictures

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Public

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Templates

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59

Videos
```

Don't worry about what the details mean as we will cover them in upcoming sections, but from the output above, we get a lot more details about the same files.

That brings us to a fundamental concept about commands. Most commands in Linux uses what we call *arguments. Arguments* are additional options used to modify the behavior of a command. (Case in point).

In some cases, options and arguments can mean two different things. For example, a command to manage wireless devices may take the device name as an option and another argument to modify its operations.

The general syntax for arguments and options is:

command -option argument

Most commands have arguments, including a single letter preceded by a dash (-) symbol such as ls -l. These arguments can also be chained together using a single dash, as shown in the example below:

```
ubuntu@VSALEM:~$ ls -al

total 12

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 15:49.

drwxr-xr-x 1 root   root   4096 Nov 2 11:15 ..

-rw------- 1 ubuntu ubuntu 996 Nov 2 15:51 .bash_history

-rw-r--r-- 1 ubuntu ubuntu 220 Nov 2 11:15 .bash_logout

-rw-r--r-- 1 ubuntu ubuntu 3771 Nov 2 11:15. bashrc

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 11:15. landscape

-rw-r--r-- 1 ubuntu ubuntu    0 Nov 2 11:15. motd_shown

-rw-r--r-- 1 ubuntu ubuntu 807 Nov 2 11:15. profile

-rw-r--r--   1   ubuntu   ubuntu            0   Nov   2   11:36.
sudo_as_admin_successful

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Desktop

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Documents

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Music

---------------TRUNCATED OUTPUT-----------------------------------
```

With the above command, the ls command gets modified using -al argument to list files in a specific format and include hidden files and directories.

You can also modify the sorting order of files and directories using the –reverse argument.

```
ubuntu@VSALEM:~$ ls -al --reverse

total 12

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Videos

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Templates

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Public

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Pictures

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Music

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Documents

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 2 12:59 Desktop

-rw-r--r--  1  ubuntu  ubuntu      0  Nov  2  11:36.
sudo_as_admin_successful

-rw-r--r-- 1 ubuntu ubuntu 807 Nov 2 11:15. profile

----------------------TRUNCATED  OUTPUT----------------
```

As you can see, the file listings order is now in reverse, listing them in an alphabetical-descending order.

Do you remember what we mentioned about how a single command can take up an entire chapter? The Ls is one of those commands.

Here, however, we shall not delve deep. Check out the ls manual to learn more.

https://man7.org/linux/man-pages/man1/ls.1.html

Instead, we shall list some of the ls most common arguments, which include:

1. -a or --all – This argument lists all files and directories, including hidden ones (beginning with a dot) not listed under a normal ls command.
2. -t – the -t argument sorts files by the date of modification.
3. -d or --directory - This argument in conjunction with -l argument lists details about the directory itself and not its contents.
4. -r or --reverse – As we have seen, the –reverse command lists files in reverse order. In a normal ls command, files and directories get displayed in an alphabetical-ascending order.
5. -l – Used to list files and directories in a long format.
6. -F or --classify – The classify argument places a character indicator at the end of a file or directory. For example, it will append a forward slash (/) to a directory.
7. -h or --human-readable – The -h argument, used in conjunction with the -l argument, shows the size of files and directories in a human-readable format. i.e., In kilobytes, megabytes, gigabytes, and so on instead of bytes.
8. -S – the -S argument sorts the files and directories by their file size.

Let us talk about the details we get from using ls -l command.

We saw something similar to this.

```
ubuntu@VSALEM:~$ ls -lahF

total 12K

drwxr-xr-x 1 ubuntu ubuntu 4.0K Nov 2 15:49./

drwxr-xr-x 1 root   root   4.0K Nov 2 11:15 ../

-rw-------   1   ubuntu   ubuntu   996   Nov   2   15:51

.bash_history

-rw-r--r--   1   ubuntu   ubuntu   220   Nov   2   11:15

.bash_logout

-rw-r--r-- 1 ubuntu ubuntu 3.7K Nov 2 11:15 .bashrc

-rw-r--r--   1   ubuntu   ubuntu         0   Nov   2   11:36

.sudo_as_admin_successful

drwxr-xr-x 1 ubuntu ubuntu 4.0K Nov 2 12:59 Desktop/

drwxr-xr-x 1 ubuntu ubuntu 4.0K Nov 2 12:59 Documents/
```

Let us examine one file and see what it means.

1. drwxr-xx-x – The first field indicates the file type and the access rights to that file. The first character in the first field indicates the file type, and the rest used to show access rights. Among various file types, a leading dash in the first character indicates a regular file while a d indicates a directory. The next three characters show the access rights of the file owner. The next three characters in the second field indicate the access rights for the filegroup members, while the last characters in the third field show access rights for everyone else.

Note: Details about what that means are in the users, groups, and permissions chapter.

2. 1 – The next is a number used to indicate the file's number of hard disks. We shall discuss file links in upcoming chapters.
3. ubuntu – The next field we see is the username of the owner of the file. Because my username is called Ubuntu, the owner is hence Ubuntu. Try listing /usr directory and see the difference
4. ubuntu – This field shows the group owner of the file. We will cover users and groups in later chapters.
5. 4.0k – This shows the size of files in a human readable format. If you do not use the -h argument, the size will appear shown in bytes.
6. Nov 2 11:15 – The next is the date and time of file modification.
7. .bashrc – This indicated the name of the file.

Section 4

Understanding the Filesystem

Now that we have learned how to explore and navigate the filesystem, we can move to the next part: understanding the organizational structure of the Linux filesystem.

We will start by learning how to view file contents. Then we shall move to how Linux organizes everything, including where applications are stored, boot configuration, users' directories, and more, and then finish off with symbolic and hard links.

As you may recall from earlier chapters, we mentioned that Linux does not require file extension to determine files and filetypes. For example, where a normal document would have a file extension like *.pdf, *.odt, or *.doc , Linux does not require that.

In this section, we will learn how to view the contents of file using the file command.

To start, create a text file using the command:

```
ubuntu@VSALEM:~$ echo "Hello world!"> myfile.txt
```

At the moment, don't worry about what the commands mean. Next, use the file command to see the file content.

```
ubuntu@VSALEM:~$ file myfile.txt

myfile.txt: ASCII text
```

Linux then tells you that myfile.txt is an ASCII text file, which is true. It's good to note that there are numerous filetypes. In Linux, you will often come across the phrase, "In Linux, everything is a file," which is true. In Linux, even directories are types of files.

The less command

Let us take on another fun Linux command, the less command. The less command is a tool that allows you to view the contents of a file. Do not confuse this with the file command that shows the type of file.

In Linux, many files are in raw ASCII text, such as source code files, bash scripts, configuration files, and much more. Using the less command allows us to view and examine what is in these files. In the next section, we will learn how to use various tools, including vi and nano , to edit files but for now, let us view what is in them.

The general syntax for the less command is:

```
less <filename>
```

Using the less command, we can view the content of file and scroll up and down using up-down arrow keys. If the files' contents are larger than a single page determined by the terminal window, we can use space to navigate to the next page. Once done with the file, we use the Q key to quit from the file.

For example, use the less command to view a file used to store sources for install packages.

```
ubuntu@VSALEM:~$ sudo less /etc/apt/sources.list
```

The less command works very well when used in combination with keyboard commands. The following are some of the keyboard shortcuts to use while working with less.

1. Spacebar – Used to scroll to the next page of a file.
2. Up Arrow – used to scroll up one line
3. Down Arrow – Scrolls down one line at a time.
4. Q – Quits the less commands
5. H – used to display the less help menu.
6. B – used to scroll to the previous page.
7. G – Scrolls to the end of the file.
8. /characters – used to Search forward to the next occurrence of a string.

The less command is an improvement of its predecessor command known as less. Try running the command:

```
ubuntu@VSALEM:~$ whatis less
```

less (1)   - opposite of more

The Linux Filesystem

The Linux filesystem is probably the most important section of this book.

Hence, the book will give you a deep understanding of the Linux filesystem, such as where the root directory is, tec.

Once you understand the FS, you will be in a position to find more commands, read, write, execute files, and so much more. Understand this part, and you can work with any Linux distro like a master.

In most cases, the layout of the Linux filesystem is the same across all Linux distribution. Although this is not always the case in all distributions, in most cases, the structure is similar and implemented by the *Linux Filesystem Hierarchy Standard.*

https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html

As we explore the filesystem and understand what lies beneath this incredible operating system, use your device to these directories practically, use the file command to examine a fil, and even use the less command to see what a certain file contains.

Linux Directory Organization

Here are the Directories found in Linux systems and what each directory is reserved to include:

1. / - This is the root directory of a Linux system where everything starts.
2. /bin – The bin directory stores or contains binaries required for the system to boot and run.
3. /boot – boot directory contains the Linux kernel, RAM disk image, and the required bootloader. You will find files such as /boot/grub/grub.conf or menu on a system that uses grub as

the bootloader .lst, Kernel images such as /boot/vmlinuz etc.

4. /dev – Dev directory contains device nodes. This is a special kind of directory used by the kernel to store a list of devices it recognizes.
5. /etc – etc is one of the most important directories in Linux systems. It contains a list of system-wide configuration files. /etc directory can also contain a collection of shell scripts used to launch services during system boot. Almost every file in the /etc directory is plain text. Common files found in this directory include /etc/passwd that has a list of user accounts, /etc/resolv.conf that has the DNS resolver configuration, /etc/crontab that has a list of startup operations.
6. /home – The home directory has specific directories for each user to prevent users from tampering with other directories. However, this is not always the case, and you may find special cases where this does not happen.
7. /lib – Lib folder contains a collection of shared library files used by the core Linux system. Think of dll directories in Windows systems.
8. /lost+found – This directory contains all devices and partitions formatted in a Linux filesystem such as ext4 or ext3. This directory's purpose is to work in case of a partial recovery from a filesystem corruption event. In most cases, unless something terrible happens to your system, this directory is usually empty.
9. /media – We mentioned the media directory in earlier chapters. It is mainly used by modern Linux systems to store mount points for removable devices mounted automatically.
10. /mnt – the /mnt directory holds devices mounted manually.

11. /opt – the /opt directory holds third party software installed on the system. If you are using commercial software, you will most likely find it here.

12. /proc – This is a special kind of directory and acts as a filesystem. It is not a real filesystem in the way files are stored in the disk. The /proc directory acts as a virtual filesystem maintained by the Linux kernel. It has files that allow a view of the kernel. Files stored in the /proc directory are readable. They give a picture of how the kernel manages the system. For example, to view the CPU information, you can use the command less /proc/cpuinfo.

13. /root – This directory contains the files for the root user. It is the official root user home directory.

14. /sbin – The /sbin directory is used by the kernel to store system binaries. System binaries are critical programs used to carry out vital tasks on the system. Linux mainly reserves them for superuser accounts.

15. /tmp – this directory stores temporary or transient files generated by various programs. This directory gets cleared upon a reboot.

16. /usr – This is another important directory in Linux. It contains programs and support files used by all users. This is one of the largest directories in the Linux system and contains thousands of files.

17. /usr/bin - /usr/bin This one holds the executable programs that have been installed by your Linux distribution.

18. /usr/lib – Contains shared libraries for programs stored in /usr/bin directory.

19. /usr/local – This directory contains programs not included by a specific Linux distribution by default but is available for system-wide use. For example, programs

compiled by users get stored in this directory. Although this directory exists by default, it remains empty until privileged users add files to it.

20. /usr/sbin – Used to store system admin programs.

21. /usr/share – The /usr/share directory contains shared data used by programs in /usr/bin directory. You will find some configuration files here, icons, themes, cursors, sound config files, and more.

22. /usr/share/doc – This directory stores documentation files for most packages installed in the system.

23. /var – This directory stores data that constantly changes, such as databases, spool files, mail, and more. All other directories in Linux except /tmp and /home are static.

24. /var/log – As the name suggests, the /tmp/log directory stores log files for various system operations. This directory requires constant monitoring as it contains critical log files for the system. You may require superuser privileges to view log files.

Now that we have explored the world of Linux FS, we can learn how to work with files and directories. How we can copy, move, delete, rename, etc. files and directories. In the next chapter, we are going to get hands-on with file operations.

Section 5:

File and Directory Manipulation in Linux

This section shall discuss the commands that allow us to work with Linux files and directories.

File and directory operations are very common when working with computers, and thus, the commands in this section are essential.

As we go through the chapter, you will discover that it is easier to move, rename, delete, and do file operation using a graphical file manager. However, if you want more control and flexibility over what you carry out, the command line provides that power.

Let us get started:

Creating Directories – mkdir

The first command we are going to look at is the mkdir command. The mkdir command creates directories in the current or specified workspace. Here is how the mkdir command works:

mkdir directoryName

Although it may seem simple enough, the mkdir is pretty powerful. You can create more than one directory in a single command as:

mkdir dir1 dir2 dir2 dir4 dir5…dir(n)

Once the command gets executed, it creates all the directories passed above as:

```
ubuntu@VSALEM:~$ mkdir dir1 dir2 dir3 dir4

ubuntu@VSALEM:~$ ls -l --reverse

total 0

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 4 14:20 dir4

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 4 14:20 dir3

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 4 14:20 dir2

drwxr-xr-x 1 ubuntu ubuntu 4096 Nov 4 14:20 dir1
```

Copying Files and Directories – cp Command

The next command is the cp command used to copy files and directories from one location to another within a Linux filesystem.

The cp commands support two syntaxes:

cp file1 file2 or cp dir1 dir2

This syntax copies file1 or dir2 to file2 or dir2. This comes in handy when working with single items or a single directory.

The next syntax for the cp command is:

```
cp item1, item2 item3 item4…item(n) directoryName
```

In this case, the cp command copies all the specified files from the current directory to the specified directory.

The cp command has a wide selection of options that you can use to manipulate how and what the command does. Although we will not discuss all of them, here are options worth mentioning.

1. -a or --archive – Using the cp command using the --archive option allows you to copy files and directories with all their respective attributes and permissions. Without the argument, the copy command uses the permissions of the user carrying out the copy command. For example, if the ubuntu user owned the copied file, and root performed the copy, root owns the file. Be careful with this.
2. -i or --interactive – The interactive argument enables the cp command to prompt the user before overwriting a file. By default, cp will automatically and silently overwrite a file.
3. -v or --verbose – As the name suggests, Verbose suggests shows informative messages as each file and directory gets copied.
4. -r or --recursive – Allows the cp command to recursive copy files and directories. This command gets plenty of use and is an almost-must when copying directories.
5. -u or --update – The update argument is very effective. It tells the cp command when copying files from one directory to another, copy files that are newer than existing files in the destination folder, or files that do not exist in the destination folder.

Moving, Renaming Files and Directories – mv command

The next command in the file operations section is the mv command. The mv command moves or renames files depending on the syntax of the command used.

Let us learn how to work with mv.

The first syntax for the mv command is similar to the cp command.

```
mv file1 file2
```

Using the syntax above, you move file1 into file2. If file2 exists, it gets overwritten with the contents of file1. If it does not exist, the file2 gets created, and file1 ceases to exist. That is the basic concept behind renaming a file.

mv file1 file2…file(n) directory

When we use the syntax above, the mv command moves all the files specified to the directory. For this to happen, the directory must exist.

Since mv and cp commands share a basic mode of operation, they share similar options that include:

1. -i or --interactive – Will prompt the user before overwriting files.
2. -v or --verbose – Displays informative messages as the file moving process continues.
3. -u or --update – Move files that do not exist or newer than existing corresponding files.

Deleting Files and Directories – rm Command

The rm command deletes files and directories. The general syntax for the command is:

rm file2 or directory1…file(n) or directory(n)

Depending on how and where you use the command, the rm command can be very dangerous. Linux does not offer an undelete command, and once a delete action occurs using the rm command, it is gone! Be careful when working with rm .

The rm command also supports various arguments that include:

1. -f or --force – This option helps ignore nonexistent files and arguments and never prompt
2. -i or --interactive – Prompts the user before deleting any existing files. Cannot be used along with the --force argument.
3. -v or --verbose – Shows informative messages as the deletion process gets carried out.

4. -r or --recursive – Recursively deletes files and directories. That means if the specified directory contains subdirectories, remove them as well.

We discuss wildcards shortly because they are helpful when working with files and directories.

NOTE: You will often find a prank played on Linux beginners on the internet telling you to execute the command cd / && sudo rm -rf *

As we continue, you will learn what the commands do but do not run the command as this will break your entire system.

Wildcards

Now that we have covered file operation commands. Let us discuss a key feature that makes these commands very powerful and better than using a GUI file manager, although applicable to some file managers such as *Nautilus, Dolphin,* or *Caja* .

The shell utilizes filenames intensively and provides a way to specify a group of filenames in a single command. These are in the form of special characters known as *wildcards,* also known as *globbers.*

Using *wildcards,* also known as *globbing,* allows you to select a group of filenames based on patterns simultaneously. The following are common wildcard supported on Linux systems.

1. * - The asterisk (*) wildcard finds and matches any characters
2. ? – Finds and matches any single character
3. [characters] – Finds and matches any characters that are members of the set characters.
4. [! characters] – Finds and matches any characters that are not members of the specified set.
5. [[:class]] – Finds and matches any characters that are members of the set class. Commonly used character classes include:

a. [:alnum:] – Using alphanumeric characters as the class.
b. [:alpha:] – alphabetical character class.
c. [:lower:] – lowercase letters character class.
d. [:upper:] – Uppercase letters character class.
e. [:digit:] – numeral character class.

Wildcards, therefore, provide a convenient way to create customized selection criteria for filenames.

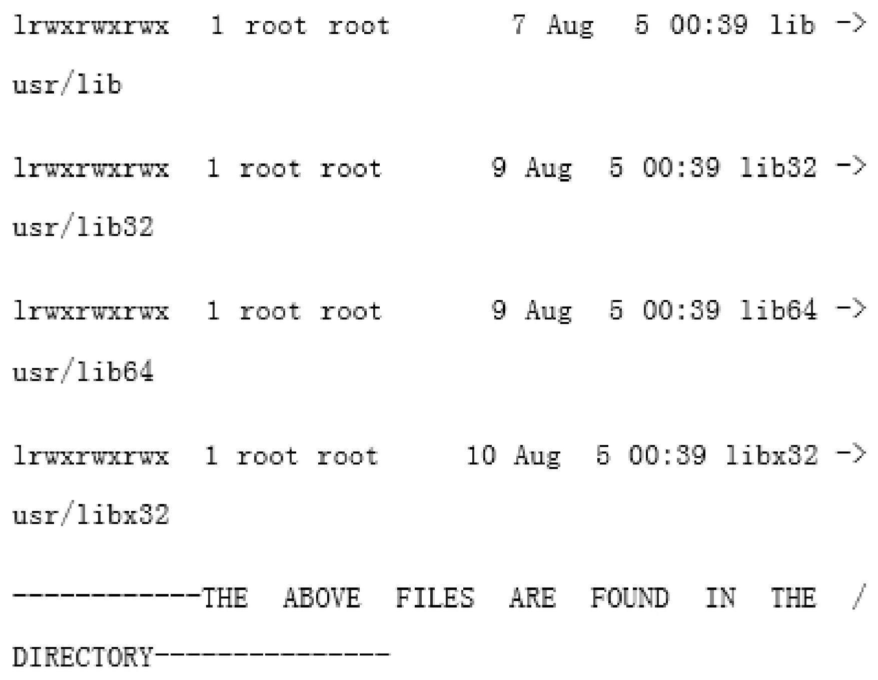
Here are some examples of wildcard use cases and their corresponding matches.

1. * - Everything including files and directories.
2. a* - Any file or directory beginning with the letter a.
3. a?? – Any files or directories beginning with the letter a and followed by exactly two characters.
4. Log.[0-9][0-9][0-9] – Any files or directories beginning with the word log followed by exactly three numerical values.
5. [[:upper:]]* – Any files or directories beginning with an uppercase letter.
6. a*.txt – Any files or directories beginning with the character a followed by any characters but ending with txt characters.
7. [abc]* - Any files or directories beginning with either the letter a, b or c.
8. [![:lower:]] – Any files or directories not beginning with a lowercase.
9. *[[:digit:]txt] – Any files or directories ending with a digit or the characters txt.

Wildcards can work with all of the commands that accept filenames such as cp, mv, rm, etc. Now you can see why the command sudo rm -rf * on the / directory is a terrible idea. We will learn more about sudo in upcoming chapters.

Soft and Hard Links

In Linux, you will sometimes come across a directory that may look something like this:



```
lrwxrwxrwx  1 root root       7 Aug  5 00:39 lib ->
usr/lib

lrwxrwxrwx  1 root root      9 Aug  5 00:39 lib32 ->
usr/lib32

lrwxrwxrwx  1 root root      9 Aug  5 00:39 lib64 ->
usr/lib64

lrwxrwxrwx  1 root root     10 Aug  5 00:39 libx32 ->
usr/libx32

------------THE  ABOVE  FILES  ARE  FOUND  IN  THE  /
DIRECTORY---------------
```

You will notice that the directory first letter in the listing is an l instead of a regular letter d for directories. It also seems to contain two directories in one. That is a special kind of file in Linux known as a soft link, symlink or *symbolic link* . Linux allows file referencing by multiple names.

From the command above, we see a symbolic link such a libx32 that points to the directory /usr/libx32 . Hence, any program looking for the libx32 directory will get the /usr/libx32 directory.

Creating Links

To create symbolic links in Linux, we use the ln command. The general syntax is:

ln file link

The above syntax creates a hard link.

ln -s item link

That creates a soft link for the passed item. The passed item can be either a directory or a file.

Hard links vs. Soft Links

There is no significant difference between hard and soft links. The main difference is how the two reference the files.

Hard links were the default ways of creating links in Linux, while symbolic links are more modern, and by default, every file has a hard link that provides a name for the file.

With a hard link created, an additional directory entry for the file also gets created. Here are the main features of hard links:

1. They are not used to reference directories.
2. They cannot reference files outside their filesystems. This means a hard link cannot reference a file in another partition than it residential location.

A hard link is indistinguishable from the main file. Contrary to a directory list holding a symbolic link, a directory list that has a hard link does not show any special indication of the link. Once a hard link gets removed, the link gets deleted, but the file’s main contents don’t get removed. This means that space doesn’t get deallocated until after file deletion.

Symbolic links are the newer and more improved version of hard links. Their development aim was to help overcome some of the limitations that come with hard links.

Basically, symbolic links work by executing a special file containing a text pointer to the referenced directory or file. They mainly work similarly to Windows shortcuts.

A symbolic link and the main file referenced are very hard to distinguish. Symbolic links do not delete files once it gets deleted. However, if the file gets removed before the link gets removed, the symbolic link becomes *broken.* You will encounter broken symbolic links indicated by red using the ls command.

NOTE: We have covered a lot of information in this section alone. It may take a while to master (not memorize) the commands and their arguments, and the best way to learn is to practice. Play around with the commands until you get them.

Section 6

Commands for Working With Commands

Up to this point, we have discussed a wide range of commands, with each command having a wider range of options that are different or performed differently.

In this chapter, we shall simplify these commands and even write unique commands.

Commands: Simple definition

Before we discuss commands that help us work with other commands, let's understand commands a bit better. What is a command, really?

As mentioned at the beginning of the book, commands are texts or characters you type in a terminal for the shell to execute. The typed commands reference something within the Linux fs.

For example, a command like ls is a collection of code that finds files and directories—the process is not as direct as it sounds. Hence, a command can reference four types of things.

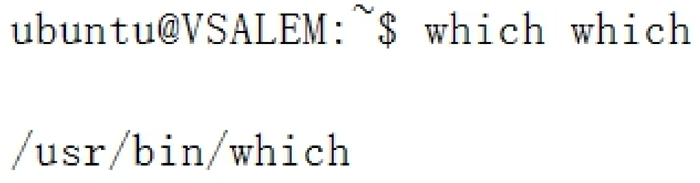
1. A shell function – *shell functions* refer to small *shell scripts* merged in the Linux environment.
2. An executable program – A command can be a compiled binary like files found in /bin and /usr/bin directories. Programmers can compile these binaries in various languages such as C/C++, Python, Perl, Ruby, etc.
3. Built-in shell command – These are a type of commands built-in into the shell—also called *shell built-ins.*
4. An alias – a command can also refer to an alias or an alternative name for a command. For example, you can create a command to show free memory from free to fmem .

Identifying Commands

Identifying commands are commands used to identify what other commands are and what actions they perform in the system. They include:

Show executable location – which command

In instances where there are multiple versions of an executable program installed on a server, we can use the which command to locate an executable file as follows:



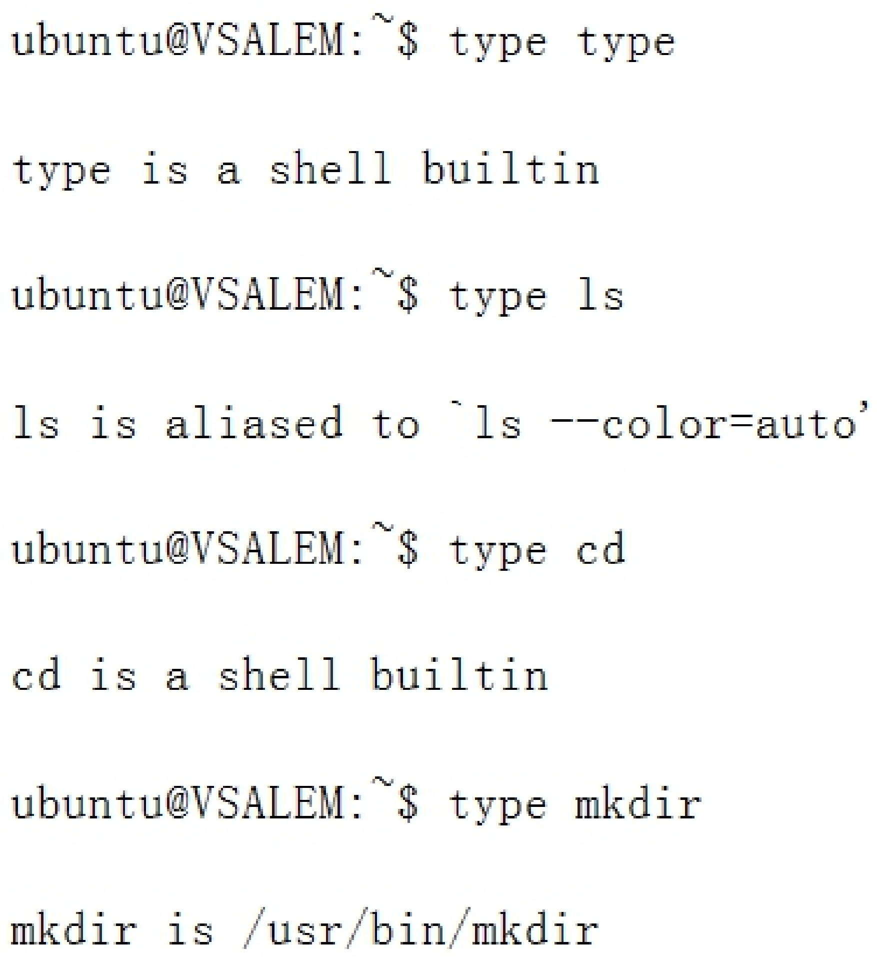
```
ubuntu@VSALEM:~$ which which

/usr/bin/which
```

Which command only works on executable programs, not on built-in shell commands. If you run the which on a command like cd , you will get an error:

Display command type – type command

The type command displays the type of command the shell is going to execute. The types can include the ones we mentioned above.

The type command works like this:



```
ubuntu@VSALEM:~$ type type

type is a shell builtin

ubuntu@VSALEM:~$ type ls

ls is aliased to `ls --color=auto'

ubuntu@VSALEM:~$ type cd

cd is a shell builtin

ubuntu@VSALEM:~$ type mkdir

mkdir is /usr/bin/mkdir
```

From the commands above, we can see that the command cd is a shell built-in, mkdir is an executable located in /usr/bin/mkdir . A command like ls is really an alias for the ls command with --color=auto argument enabled.

Displaying Commands Help and Documentation

Once we know what a command is and its location, we can find its documentation stored in the system's manual database.

The help command helps you find out about the options and arguments the command accepts. In most cases, the help command only works on shell built-in commands. For example:

```
ubuntu@VSALEM:~$ help help

help: help [-dms] [pattern ...]

    Display information about builtin commands.

    Displays brief summaries of builtin commands.  If PATTERN is

    specified, gives detailed help on all commands matching PATTERN,

    otherwise the list of help topics is printed.

    Options:

      -d        output short description for each topic

      -m        display usage in pseudo-manpage format

      -s        output only a short usage synopsis for each topic matching

                PATTERN

    Arguments:

      PATTERN   Pattern specifying a help topic



    Exit Status:

    Returns success unless PATTERN is not found or an invalid option is
given.
```

Although the help command provides options and how they work, it is in no way a tutorial. However, it comes in handy when you want to learn more about a command.

Another popular mode of the help command is using it as an argument using the --help syntax.

Most executable programs in Linux provides this argument to allow you to find a command’s usage information. For example:

```
ubuntu@VSALEM:~$ ls --help

Usage: ls [OPTION]... [FILE]...

List  information  about  the  FILEs  (the  current
directory by default).

Sort entries alphabetically if none of -cftuvSUX nor -
-sort is specified.

Mandatory arguments to long options are mandatory for
short options too.

  -a, --all                  do not ignore entries
starting with .

  -A, --almost-all           do not list implied . and
..

      --author               with -l, print the author
of each file

  -b, --escape               print C-style escapes for
nongraphic characters

      --block-size=SIZE      with -l, scale sizes by
SIZE when printing them;
```

```
                                e.g., '--block-size=M';
see SIZE format below
  -B, --ignore-backups          do not list implied
entries ending with ~
  -c                            with -lt: sort by, and
show, ctime (time of last
                                  modification of file
status information);
                                with -l: show ctime and
sort by name;
                                  otherwise: sort by
ctime, newest first
  -C                            list entries by columns
      --color[=WHEN]            colorize the output; WHEN
can be 'always' (default
                                if omitted), 'auto',
or 'never'; more info below
  -d, --directory                 list directories
themselves, not their contents

--------------------------OUTPUT TRUNCATED---------
-------------------------------
```

Not all commands support the --help argument. Some support -h and others use manual pages. However, try it to find out if it works.

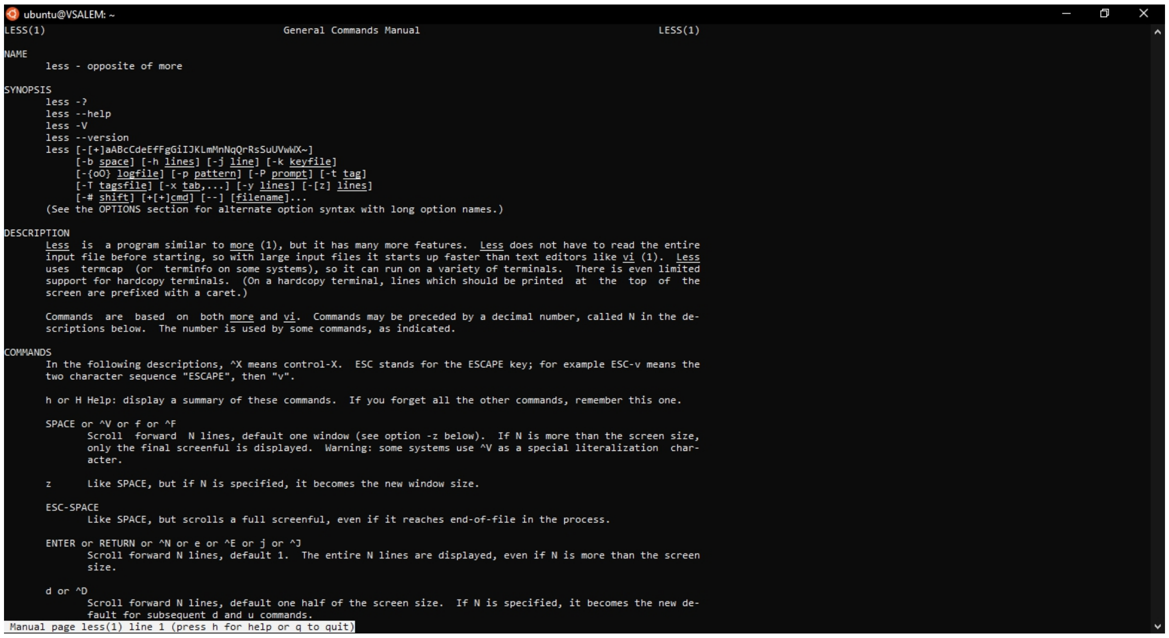
The most common way to find information about commands is by using *manual pages.* Manual pages are formal pieces of documentation for various commands. *Manual pages* are commonly known as *man pages* and are accessible using a special command called man .

The general syntax for the command is:

man program

Where "program" represents the command that you need documentation for. Although manual pages contain different formats and styles, they mainly contain a title, a synopsis of the command's syntax, and a description of what the command does. It may also include a listing and a description of the command's options. Manual pages are sort of like a reference and not a guide.

Manual pages use the less command to navigate the documentation. Try to view the manual pages for the less command.



```
ubuntu@VSALEM: ~
LESS(1)                          General Commands Manual                         LESS(1)

NAME
       less - opposite of more

SYNOPSIS
       less -?
       less --help
       less -V
       less --version
       less [-[+]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]
            [-b space] [-h lines] [-j line] [-k keyfile]
            [-{oO} logfile] [-p pattern] [-P prompt] [-t tag]
            [-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
            [-# shift] [+[+]cmd] [--] [filename]...
       (See the OPTIONS section for alternate option syntax with long option names.)

DESCRIPTION
       Less  is  a program similar to more (1), but it has many more features.  Less does not have to read the entire
       input file before starting, so with large input files it starts up faster than text editors like vi (1).  Less
       uses  termcap  (or  terminfo on some systems), so it can run on a variety of terminals.  There is even limited
       support for hardcopy terminals.  (On a hardcopy terminal, lines which should be printed  at  the  top  of  the
       screen are prefixed with a caret.)

       Commands  are  based  on  both more and vi.  Commands may be preceded by a decimal number, called N in the de-
       scriptions below.  The number is used by some commands, as indicated.

COMMANDS
       In the following descriptions, ^X means control-X.  ESC stands for the ESCAPE key; for example ESC-v means the
       two character sequence "ESCAPE", then "v".

       h or H Help: display a summary of these commands.  If you forget all the other commands, remember this one.

       SPACE or ^V or f or ^F
              Scroll  forward  N lines, default one window (see option -z below).  If N is more than the screen size,
              only the final screenful is displayed.  Warning: some systems use ^V as a special literalization  char-
              acter.

       z      Like SPACE, but if N is specified, it becomes the new window size.

       ESC-SPACE
              Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the process.

       ENTER or RETURN or ^N or e or ^E or j or ^J
              Scroll forward N lines, default 1.  The entire N lines are displayed, even if N is more than the screen
              size.

       d or ^D
              Scroll forward N lines, default one half of the screen size.  If N is specified, it becomes the new de-
              fault for subsequent d and u commands.
 Manual page less(1) line 1 (press h for help or q to quit)
```

Manual Pages Organization

Manual pages are amazing tools when working with commands. They get stored in the manual database organized in a specific format covering executable commands, user commands, system administration commands, file formats, etc.

Below is an outline of the layout of the manual pages.

1. 1 – The first section of the manual pages contains documentation for user commands.
2. 2 – The second section is features kernel system calls programming interfaces.
3. 3 – The third section contains C library's programming interfaces documentation.
4. 4 – The fourth section is for special files. This includes drivers and device nodes.
5. 5 – Five is for file formats documentation.
6. 6 – The sixth section is for games and other enjoyments such as wallpapers and screensavers.
7. 7 – Seven is for miscellaneous

    a. 8 – The eighth section contains documentation for system administration commands.

Sometimes you may have to search in a specific documentation section to find the content you are looking for. For example:

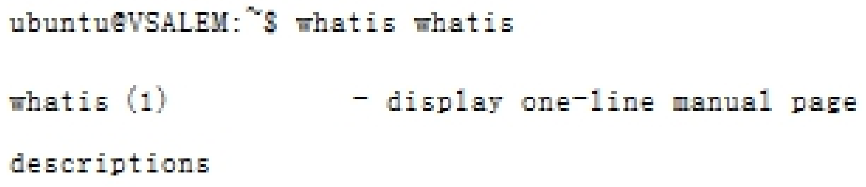
```
ubuntu@VSALEM:/etc$ man 5 sudoers
```

The command above shows the documentation showing file formats for the sudoers file.

Display Brief Description – whatis command

The whatis command is a handy command. It displays the name of the command and one-line description from the man pages matching a specific keyword.

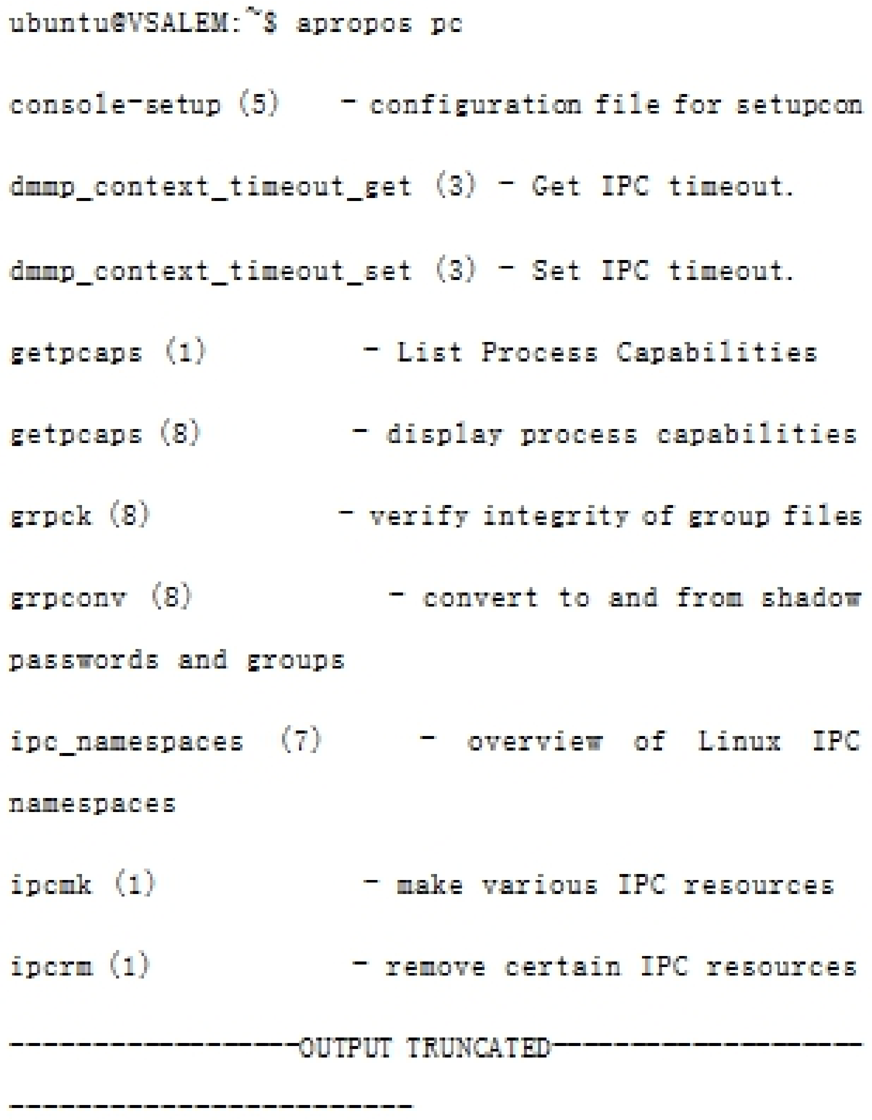
For example:



```
ubuntu@VSALEM:~$ whatis whatis

whatis (1)           - display one-line manual page

descriptions
```

Display Appropriate Commands – Apropos

Manual pages allow you to search for a match based on the passed term. Using the command apropos , you can search a similar command from the manual pages. For example:



```
ubuntu@VSALEM:~$ apropos pc

console-setup (5)    - configuration file for setupcon

dmmp_context_timeout_get (3) - Get IPC timeout.

dmmp_context_timeout_set (3) - Set IPC timeout.

getpcaps (1)          - List Process Capabilities

getpcaps (8)          - display process capabilities

grpck (8)            - verify integrity of group files

grpconv (8)            - convert to and from shadow

passwords and groups

ipc_namespaces  (7)      -  overview  of  Linux  IPC

namespaces

ipcmk (1)             - make various IPC resources

ipcrm (1)             - remove certain IPC resources

-----------------OUTPUT TRUNCATED------------------

------------------------
```

You can also substitute the apropos command with man -k argument.

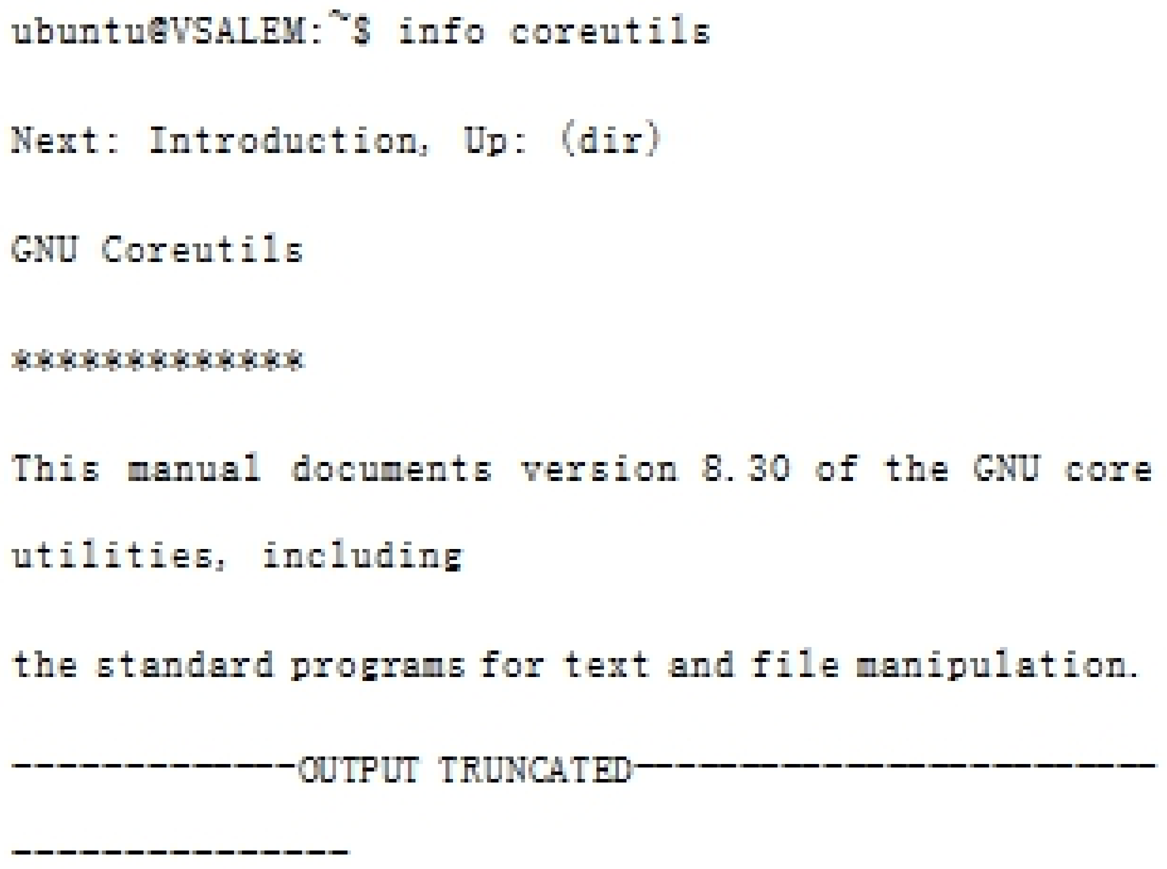
Another common command you can use to find command help and documentation is the info command. It is an alternative to the man command but provides hyperlinks to help navigate to specific sections.

The info program reads info files, structured like trees into individual nodes, each encompassing a single topic. Info files have hyperlinks you can use to move you from one node to the other. Hyperlinks are easy to identify because of their leading asterisk. You can activate them by placing the cursor on it, then hitting the ENTER key.

For example:

```
ubuntu@VSALEM:~$ info ls
```

All the commands we have up to this point belong to the coreutils package or the GNU Project. To find more details about residence and documentation, use the command:



```
ubuntu@VSALEM:~$ info coreutils

Next: Introduction, Up: (dir)

GNU Coreutils

*************

This manual documents version 8.30 of the GNU core

utilities, including

the standard programs for text and file manipulation.

------------OUTPUT TRUNCATED---------------------

---------------
```

Many of the software packages you have installed on your system will have documentation files stored in the directory /usr/share/doc . Most get stored in plaintext format, but some are HTML and viewable with a web browser.

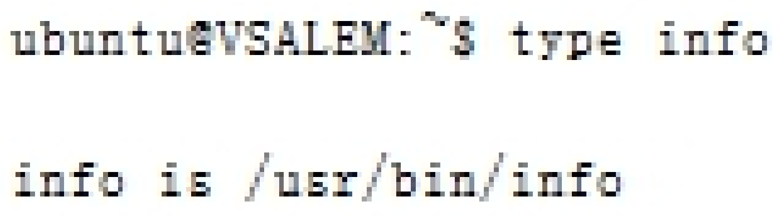
Some files will have a .gz extension, indicating files compressed using the gzip compression program. The gzip package has a special less version called zless that displays the information contained in gzip-compressed text files.

Manual pages can be very good as references for a wide range of commands. However, as we mentioned, they are not tutorials. If you want to test how crazy manual pages are, try manual pages for bash. Do not get frustrated, though; keep learning: you will understand what it all means.

Creating Custom Commands

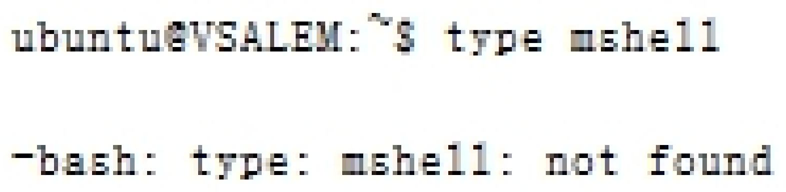
This subsection covers how to create commands using the alias command. The alias command allows you to come up with better names to substitute a command or a single command to represent a sequence of commands.

Before we begin, the first task is to find out an appropriate name for our command and ensure no other critical function is using it.



```
ubuntu@VSALEM:~$ type info

info is /usr/bin/info
```

Unfortunately, info seems taken, and we cannot use it. Let us try mshell :



```
ubuntu@VSALEM:~$ type mshell

-bash: type: mshell: not found
```

That one is available to use. We use the alias command, followed by a command or a sequence of commands we want to use.

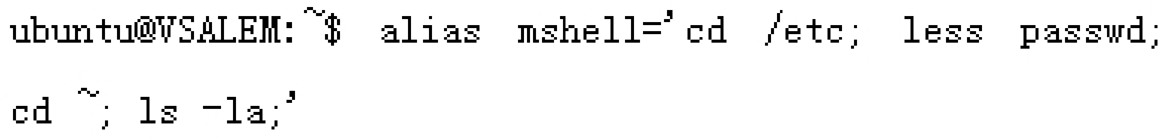
For example, to navigate to the /etc directory, read the passwd file, navigate back home, and finally list the directories, we can combine all the commands in one line using a shell trick shown below:

cd /etc; less passwd; cd ~; ls -la;

Using alias , we can create a single command as shown below: The general syntax for the alias command is:

alias name='string'

Pay attention to the structure of the command below:



```
ubuntu@VSALEM:~$ alias mshell='cd /etc; less passwd;

cd ~; ls -la;'
```

Immediately after the alias command, we pass the name and a string of commands we want to substitute. Once a command has been defined using the alias command, we can use it anywhere.

We can see our alias using the type command.

ubuntu@VSALEM:~$ type mshell

mshell is aliased to `cd /etc; less passwd; cd ~; ls -la;'

To remove an aliased command, use the unalias command as shown below.

ubuntu@VSALEM:~$ unalias mshell

ubuntu@VSALEM:~$ type mshell

-bash: type: mshell: not found

Granted, we avoided using an alias of an existing command because sometimes it is appropriate to do so. For example, the command ls is an alias of ls --color-auto . The following are some of the aliases defined in the environment, use the alias command without arguments to find out more.

```
alias alert=’notify-send --urgency=low -i ”$([ $? = 0 ] && echo terminal || echo error)” ”$(history|tail -n1|sed -e ’\’’s/^\s*[0-9]\+\s*//;s/[;&|]\s*alert$//’\’’)”’

alias egrep=’egrep --color=auto’

alias fgrep=’fgrep --color=auto’

alias grep=’grep --color=auto’

alias l=’ls -CF’

alias la=’  ls -A’

alias ll=’ls -alF’

alias ls=’ls --color=auto’
```

Section 7
Permissions

One of the best features of Linux and other Unix-based systems is that they are *multitasking* and *multiuser.* That means more than one user can use systems simultaneously and run more than one process.

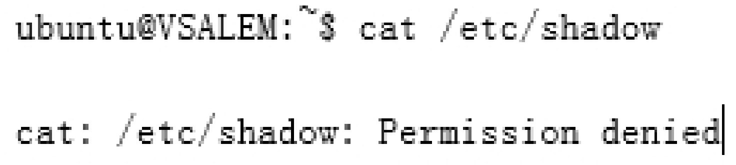
For example, a system attached to a network can have one user working on it physically and others working remotely like SSH. Features such as the X Window system allow remote users to run GUI applications remotely.

That is an amazing functionality built deep in the Linux kernel. However, to implement this, users had to be protected from each other, preventing users from interfering with files created by other users.

This chapter will look at how Linux implements protection measures like identity, file permissions, file ownership, groups, etc.

Owners, Groups, and Others

Once in a while, when exploring the Linux file system, you will encounter files that output the "Permission Denied" error when you try to read them. An example of such a file is found in /etc/shadow



```
ubuntu@VSALEM:~$ cat /etc/shadow

cat: /etc/shadow: Permission denied
```

The main reason for this type of error results from user error. Meaning, if you are running as a regular user, you may not have permission to read some files.

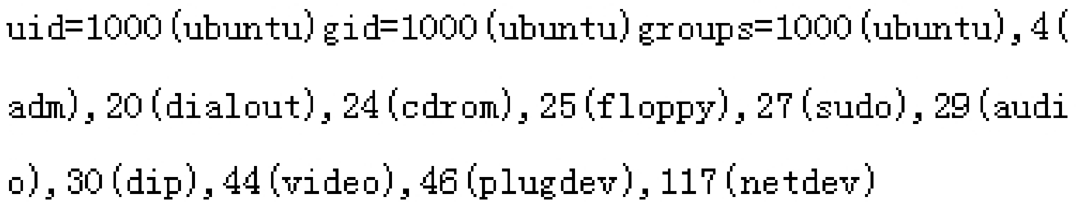
Let us discuss how this works:

Linux uses what is we call the *Unix Security model,* which describes that a user can *own* files and directories within a File system. If a user owns a specific file or directory, he/she has full control over them. The user can then belong to a *group* that consists of more than one user, all given access to files and directories by their *owners.*

Besides providing access to the groups they belong to, a user or directory *owner* can grant access rights to the *world* or *everyone* within the system.

You can find more information about your user identity by using the command id .

ubuntu@VSALEM:~$ id



```
uid=1000(ubuntu)gid=1000(ubuntu)groups=1000(ubuntu),4(
adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audi
o),30(dip),44(video),46(plugdev),117(netdev)
```

Let us dissect the output and see what it means (not all of it!)

When a user gets created, he/she gets assigned a unique value referred to as a *uid* or *unique id* value, then mapped to the user's username. The system assigns the user a *primary group id,* also called a *gid* value. The group id may also belong to other groups.

Depending on the system you run the id command on, you will get different results. For example, a system like Fedora starts numbering accounts from 500. On the other hand, Ubuntu starts at 1000.

A single user in Ubuntu can belong in more groups—shown above—because of how the Ubuntu system handles system services and device privileges.

system

devices and services.

This information gets derived from Linux text files. Accounts in Linux get stored in /etc/passwd file, while groups get stored in /etc/groups file. Upon creation of a Linux user account or group, each of these files gets modified. A file such as /etc/shadow, used to store user passwords, also gets changed.

For every user account, the /etc/passwd file defines the

user (login) name, the uid, the gid, the account’s

real name, the home directory, and the login shell.

Should you examine the contents of /etc/passwd and /etc/group , you’ll notice regular user accounts, the superuser (uid 0, accounts, and various other system users.

Read, Write, and Execute File Permissions

Access rights in Linux are defined using read, write, and execute permissions. In the previous chapters, the ls command showed this.

Let us discuss how they work and their implementation.

```
ubuntu@VSALEM:~$ ls -la

-------------------OUTPUT TRUNCATED--------------

-rw-r--r-- 1 ubuntu ubuntu    0 Nov 3 09:30 myfile.txt
```

As discussed in previous chapters, the first letter in the first block (in this case, a dash) indicates the file type. A – (dash) represents regular files, d represents directories, l symbolic links, c represents a character-special file, and b represents block devices files.

The rest nine blocks represent *file modes* that are read, write, and execute. When a directory or file has r, w, or x attribute, certain effects get applied:

1. r – Grants read permission allowing the user and group to open and read a file’s contents. If applied on a directory, a user can list the directory’s contents only when the execute attribute is active.
2. w – Permits file writing or truncating; however, the write attribute disallows file renaming and deletion. Deleting or

renaming files depends on attributes allocated to the parent directory. When you set the execute attribute, the write attribute allows the creation, deletion, and renaming of directory files.

3. x – Makes it possible to represent a file as an executable program and executed. To make files written using scripting languages executable, you must also set them as readable. On a directory, the execute attribute makes it possible to enter a directory, e.g., cd directory .

The following are examples of permission attributes set on files and directories.

1. -rwx------ - Shows a regular file with read, write, and executable permissions for the file owner only. No one else has either read, write, execute permission to the file.
2. -rw------- - Also a regular file with read and write for the file owner and no one else.
3. -rw-r--r-- - A regular file that has read and write for the file owner, the owner's group may also read the file and the world can read the file too.
4. -rwxr-xr-x – Shows are regular file with read, write, and execute permissions for the file owner, execute and read for the user's group and execute only for everyone else.
5. drwxrwx--- - Indicates a directory. The owner and user group members may enter the directory, and create, rename and delete files within the directory.
6. Lrwxrwxrwx – A symbolic link with placeholder permissions. Placeholder permissions mean that the real permissions get managed by the actual file referenced by the symbolic link.

Changing File Modes – chmod command

To change the permissions of a file or directory, we use the chmod command.

Changing file permissions is only possible for file owners and superuser accounts. Chmod supports two main ways to specify the permission

1. Octal notation
2. Symbolic notation

Symbolic Notation

Let us start with the symbolic representation of file permissions. Symbolic notations have three main parts: who the change will affect, the operation performed, and the permission set.

On who the changes will affect, letters such as u, g, o, and a are used. Here are the meanings:

- u – file owner
- g – group owner
- – Others or world
- a – all; meaning the combination of user, group and world

If no character is specified, all is assumed, and changes apply to users, groups, and the world.

The operation to be carried out gets specified in three main characters, i.e., + (plus), – (minus), and = (equals). The plus indicates that you should add permission. A minus indicates permission should be removed, and equals means that only the specified permissions should apply with others removed.

The permissions to be applied get specified in three letters, r (read), w (write), and x (execute).

Here are some symbolic notation examples:

a. u+r – Gives file owners read permission

b. u+x – Gives file owners execute permission

c. u-w – Removes write permissions for the file owner.

d. +x – Adds the execute permission for owner, group, and world.

e. u+x,go=rx – Gives files owners execute permission. It also sets read and execute group permissions—and others. Commas can separate multiple specifications

Octal Notation

Octal notation, as the name suggests, uses octal values to indicate wanted permissions patterns. You can learn more about Octal (base 8) here:

https://www.tutorialspoint.com/octal-number-system

https://en.wikipedia.org/wiki/Octal#:~:text=The%20octal%20numeral%20system%2C%20or,for%20decimal%2074%20is%201001010
.

Because every octal number digit represents three binary digits, this plots efficiently to the structure adopted when storing the file mode. The following are the octal representations and their meaning.

- 0 – no read, write, or execute permission—represented as 000 in binary and --- as a symbolic file permission block.
- 1 – Execute permission only—represented as 001 in binary and --x as a symbolic file permission.
- 2 – Read permission only—represented as 010 in binary and -w- as a symbolic file permission.
- 3 – Write and Execute permission—represent in binary notation as 011 and -wx as symbolic file permission.
- 4 – Read permission only—represented as 100 in binary and r-- in symbolic.

- 5 – Read and execute—represented in binary 101 and r-x in symbolic.
- 6 – Read and Write permission—represented as 110 in binary and rw- in symbolic notation.
- 7 – Read, write, and execute permissions. 111 in binary representation and rwx in symbolic.

We can use three octal digits to set and define the file mode for various owners and the world. For example:

```
ubuntu@VSALEM:~$ chmod 777 myfile.txt

ubuntu@VSALEM:~$ ls -l myfile.txt

-rwxrwxrwx 1 ubuntu ubuntu 0 Nov 3 09:30 myfile.txt
```

Because of passing the octal value 777, we could set read, write, and execute for all, i.e., users, groups, and everyone.

Set Default Permissions – umask command

The umask command allows you to control the default permission assigned to a file upon creation. umask uses octal notation to represent a mask of the bit removed from a file mode's attributes. For example:

```
ubuntu@VSALEM:~$ rm -rf myfile.txt

ubuntu@VSALEM:~$ umask

0022

ubuntu@VSALEM:~$ touch myfile.txt

ubuntu@VSALEM:~$ ls -la myfile.txt

-rw-r--r-- 1 ubuntu ubuntu 0 Nov 3 13:18 myfile.txt
```

We begin by cleaning any existing copies of the file using the rm -rf myfile.txt command.

Next, we executed the umask command without any arguments to view the current value that corresponds to a value 0022 (you will often find 0002 too), which is an octal notation of our mask.

Finally, we create a new instance of the file and look at the default permissions.

Sudo and su Users

The su and sudo commands are very useful while working with Linux.

Let us start with su and discuss what it does.

The su command starts a new shell session as another user. The general syntax for the command is:

su [-[l]] [user]

If you call the su command with the -l argument, the shell provides a login shell for the specified user. That means the user's environment gets loaded, and the working directory changes to the user’s home dir. If the user to login to is not specified, it defaults to the superuser or root account. It is also possible to substitute the -l argument with –

For example, to log in as root, we use the command:

```
ubuntu@VSALEM:~$ su -

Password:

root@VSALEM:~#
```

The command above prompts for the root user account’s password, and if correct, we login as the superuser account, indicated by the pound sign and the username in the shell prompt. To exit from a shell session, we use the exit command.

You can also run a single command as a superuser without having to log in as root. For example:

```
ubuntu@VSALEM:~$ su -

Password:

root@VSALEM:~# su -c 'cat /etc/shadow'

root:$6$PpH7f0r6yTKzyMlC$ZdIYZrnYyilE2/YRZoaimdsSbe3Gb

mEJoFGCJNrucclmz6gTABCVERf7WuEAZrfrUlK496s8AtSesIlGzpJ

Rk1:18572:0:99999:7:::

daemon:*:18478:0:99999:7:::

bin:*:18478:0:99999:7:::

sys:*:18478:0:99999:7:::

sync:*:18478:0:99999:7:::

games:*:18478:0:99999:7:::

----------------------OUTPUT  TRUNCATED----------------

------------------------
```

Another way to execute commands as another user is by using the sudo command. It is similar to su but provides a lot more functionalities.

The superuser can configure sudo account, allowing another user to execute commands as other users (mostly the root user).

Sudo makes it possible to restrict some users to certain commands. For example, a user working with networking and processes can operate under restriction to these commands, allowing him to do that specifically without interacting with other administrative commands.

Unlike su where a user requires a root account to log in into the shell, sudo requires a user to provide an account’s password. However, the user should be in the group known as sudoers to run commands as sudo .

Changing File Owners – chown command

To change the owner of a file or directory, we use the chown command. Chown, however, requires superuser privileges either using su or sudo .

The general syntax for the command is:

```
chown [Option]... [Owner][:[Group]] File
```

Examples:

```
  chown  root  /u           Change  the  owner  of  /u  to
 "root".

  chown  root:staff  /u   Likewise,  but  also  change  its
 group  to  "staff".

  chown  -hR  root  /u      Change  the  owner  of  /u  and
 subfiles  to  "root".
```

You may encounter systems that use the command chgrp to change the group of a file.

Changing Users password – passwd command

The last concept in this chapter and book is how to change your password and other users' passwords using superuser privileges.

To change or set a password for a specific user account, we use the command passwd .

The general syntax for the command is:

passwd [*user*]

To change the password for the current user, simply enter the command passwd without arguments, which will prompt you to enter a new password.

Depending on the system, passwd may require you to use a strong password and reject passwords that are too weak or similar to previously set passwords. To change a password for another user, use the passwd command with sudo or su .

Check manual pages for the command to learn more about the arguments available.

Conclusion

When it comes to mastering the Linux command line, the only secret is practicing, practicing, practicing some more, and learning hungrily.

This is only a beginner's guide, meaning we left many advanced concepts untouched—we have barely scratched the surface of what the Linux command line can do.

Check out the /usr/bin directory to see for yourself.

PS: I'd like your feedback. If you are happy with this book, please leave a review on Amazon.

Please leave a review for this book on Amazon by visiting the page below:

https://amzn.to/2VMR5qr