History:

1. Pre-requisites
2. What is Operating System?
3. Responsibilities of Operating System?
4. What are the Operating System Flavors available?
5. What is Linux?
6. Linux History?
7. What is Linux Foundation?
8. What is Linux Community?
9. What are Linux Distributions?
10. Installation of Linux
11. Linux Terminology
12. Linux architecture diagram

Basics:

* 1. Spinup linux machine
  2. Understand root and ec2 users
  3. Understand how to switch between users
  4. How to create directory
  5. How to delete/remove empty directory
  6. How to delete/remove non-empty directory
  7. How to create files
  8. How to delete files
  9. How to add content to files
  10. How to make file content zero bytes
  11. How to create multiple files at a time
  12. How to delete multiple files at a time.
  13. How to COPY files from one place and another place
  14. How to move files form one place to another place
  15. Difference COPY and MOVE
  16. Find command to search files and directories.

User Management and File Management:

1. Create user - useradd
2. Set password - passwd
3. Delete user -
4. Update user
5. Create group
6. Delete group
7. Update group
8. Add single user to group
9. Add multiple users to group
10. How to make user as admin
11. How to create multiple admins in a group
12. password less authentication to access the instances. (without. pem)
13. Lock user
14. Unlock user
15. Lock group
16. Unlock group
17. Find where users stored and passwords to users stored
18. Find where group stored and passwords stored.

File Administration

1. Create files using ec2-user
2. Create files using root user
3. Change the ownership of the file
4. Change the ownership of the group
5. Change the ownership of the user and group at a time
6. Permissions
7. How to add permission to file
8. How to remove permission to file
9. How to add permission to directory
10. How to remove permission to directory
11. Umask
12. SUID SGID STICKBIT (Need to work on it)
13. ACCESS CONTROL LISTS

FHS (Filesystem Hierarchy Standard)

1. **What is Operating System?**

Before moving forward, let us first know **What is an Operating System?**

An Operating System is the interface between the computer hardware and the end-user. Processing of data, running applications, file management and handling the memory is all managed by the computer OS. Windows, Mac, Android etc. Are examples of Operating systems which are generally used nowadays.

All modern computing devices including Laptops, Tablet, mobile phones, etc. comprise an Operating System which helps in the smooth working of the device.

## **What is an Operating System?**

An operating system is a collection of software between the user and their computer hardware. It controls and manages all of a computer's resources, including CPU (central processing unit) usage, memory, input and output, and network connections. An operating system also provides a UI (user interface) for users to interact with the computer and its programs. This includes the desktop, shortcut icons, file explorers, menus, taskbars, and much more.

In most situations, a user interacts with the computer through applications in order to perform everyday tasks. Applications include nearly every kind of software imaginable: web browsers, word processing software, media players, etc. These applications then communicate with the operating system, which then tells the hardware of the computer what to do. For example, if a person wishes to play a video from their hard drive, the user will interact with an application that plays the video. When the user clicks a play button, the app tells the operating system to interact with various pieces of hardware: the hard drive, where the video is stored; the display, which shows the video; the speakers, which plays the audio.

**2. What are the Responsibilities of Operating System?**

**Kernal manages the Linux IO Management, device management, file management, memory management, etc.**

* process scheduling and process management
* Memory management
* Providing security
* controlling I/O activities.
* Device Management
* Control over system performance
* Job Scheduling
* Error Detecting
* Coordinating between users and other software.

**3. What are the Operating System Flavors available?**

**Windows family**

Client based Server Based

Win xp win 2k

Win 7 win 2k3

Win 8 win 2k12

Win 10 win2k16

Win 11

**Unix Family all are server based.**

Unix

Linux

Solaris

UX is propriety OS of HP

AIX is propriety OS of IBM

Host – means always a server based (all UNIX based OS are servers)

## **4. Categories of Operating Systems**

There are many types of operating systems that fulfill various functions. They are:

1. **Batch operating system:** based on punched cards or different kinds of tape that had to be loaded into the computer, processed, then unloaded.
2. **Time-sharing operating system:** allows many users to interact with the computer at a time.
3. **Embedded operating system:** an operating system that is designed for the specific purpose of the device it is loaded onto.
4. **Multiprogramming operating system**: operating systems that can handle multiple processes at a given time.
5. **Network operating system:** an operating system on a server that allows for multiple computers to connect and share data.
6. **Distributed operating system:** an operating system that facilitates communication between other computers and itself.
7. **Multiprocessing operating system:** an operating system that utilizes multiple processors in a single computer.
8. **Real-time operating system:** an operating system that rapidly processes data in real time for use in applications and other software.

Today, the most common desktop operating systems are Windows and macOS. These operating systems are actually collections of several types of programs that handle a suite of tasks, from networking, file management, data processing, image and sound editing, and so much more. These operating systems are used in their respective platforms: Windows for personal computers, and macOS for Apple computers. Mobile devices also use their own respective operating systems.

**4. What is Linux?**

Linux is an open-source computer operating system, initially developed on and for Intel x86-based personal computers. It has been subsequently ported to an astoundingly long list of other hardware platforms, from tiny embedded appliances to the world's largest supercomputers.

Linux is a multiuser, multitasking OS

Linux started by Linus Torvalds

Linux is Open Source not like AIX, Sun Solaris, HP-UX

Linux is “case sensitive”

**5. Linux History**

Linus Torvalds was a student in Helsinki, Finland, in 1991, when he started a project: writing his own operating system kernel. He also collected together and/or developed the other essential ingredients required to construct an entire operating system with his kernel at the center. It wasn't long before this became known as the Linux kernel.

In 1992, Linux was re-licensed using the General Public License (GPL) by GNU (a project of the Free Software Foundation or FSF, which promotes freely available software), which made it possible to build a worldwide community of developers. By combining the kernel with other system components from the GNU project, numerous other developers created complete systems called Linux distributions in the mid-90’s.

The Linux Foundation is the umbrella organization for many critical open-source projects that power corporations, spanning all industry sectors. Its work today extends far beyond Linux, fostering Innovation at every layer of the software stack.

The Linux Foundation training is for the community and by the community. Linux training is distribution-flexible, technically advanced, and created with the leaders of the Linux development community.

There are three major distribution families within Linux: Red Hat, SUSE and Debian. In this course, we will work with representative members of all of these families throughout.

**6. Linux Foundation**

[The Linux Foundation](https://www.linuxfoundation.org/) provides a neutral, trusted hub for developers to code, manage, and scale open technology projects. Founded in 2000, The Linux Foundation is supported by more than 1,000 members and is the world’s leading home for collaboration on open-source software, open standards, open data, and open hardware. The Linux Foundation’s methodology focuses on leveraging best practices and addressing the needs of contributors, users, and solution providers to create sustainable models for open collaboration.

The Linux Foundation hosts Linux, the world's largest and most pervasive open-source software project in history. It is also home to Linux creator Linus Torvalds and leads maintainer Greg Kroah-Hartman. The success of Linux has catalyzed growth in the open-source community, demonstrating the commercial efficacy of open source and inspiring countless new projects across all industries and levels of the technology stack.

As a result, the Linux Foundation today hosts far more than Linux; it is the umbrella for many critical open-source projects that power corporations today, spanning virtually all industry sectors. Some of the technologies we focus on include big data and analytics, networking, embedded systems and IoT, web tools, cloud computing, edge computing, automotive, security, blockchain, and many more.

The Linux Foundation is a vendor-neutral, non-profit organization dedicated to supporting the open-source community through financial and intellectual resources, infrastructure services, events, and training.

The Foundation hosts hundreds of the world's most important open-source projects including Linux, Kubernetes, Node.js, Hyperledger, ONAP, Cloud Foundry, Yocto, GraphQL, Jenkins, and many more.

With most utilizing real-world style simulated environments that test your actual skills as well as your knowledge.

**7. Linux Community**

Suppose that, as part of your job, you need to configure a Linux file server, and you run into some difficulties. If you are not able to figure out the answer yourself or get help from a co-worker, the Linux community might just save the day!

There are many ways to engage with the Linux community:

* Post queries on relevant discussion forums.
* Subscribe to discussion threads.
* Join local Linux groups that meet in your area.

**8. Linux Distributions**

The vast variety of Linux distributions is designed to cater to many different audiences and organizations, according to their specific needs and tastes. However, large organizations, such as companies and governmental institutions, and other entities, tend to choose the major commercially-supported distributions from Red Hat, SUSE, and Canonical (Ubuntu).

CentOS and CentOS Stream are popular free (as in no cost) alternatives to Red Hat Enterprise Linux (RHEL) and are often used by organizations that are comfortable operating without paid technical support. Ubuntu and Fedora are widely used by developers and are also popular in the educational realm. Scientific Linux is favored by the scientific research community for its compatibility with scientific and mathematical software packages. Both CentOS variants are binary-compatible with RHEL; i.e., in most cases, binary software packages will install properly across the distributions.

Note that CentOS is planned to disappear at the end of 2021 in favor of CentOS Stream. However, there are at least two new RHEL-derived substitutes: Alma Linux and Rocky Linux which are establishing a foothold.

Many commercial distributors, including Red Hat, Ubuntu, SUSE, and Oracle, provide long-term fee-based support for their distributions, as well as hardware and software certification. All major distributors provide update services for keeping your system primed with the latest security and bug fixes, and performance enhancements, as well as provide online support resources.

For now, what you need to know is that this course focuses on the three major Linux distribution families that currently exist. However, as long as there are talented contributors, the families of distributions and the distributions within these families will continue to change and grow. People see a need, and develop special configurations and utilities to respond to that need. Sometimes that effort creates a whole new distribution of Linux. Sometimes, that effort will leverage an existing distribution to expand the members of an existing family.

Suppose you have been assigned to a project building a product for a Linux platform. Project requirements include making sure the project works properly on the most widely used Linux distributions. To accomplish this, you need to learn about the different components, services, and configurations associated with each distribution. We are about to look at how you would go about doing exactly that.

Complete list of Distribution available in this link:

[LWN Distributions List](https://lwn.net/Distributions/)

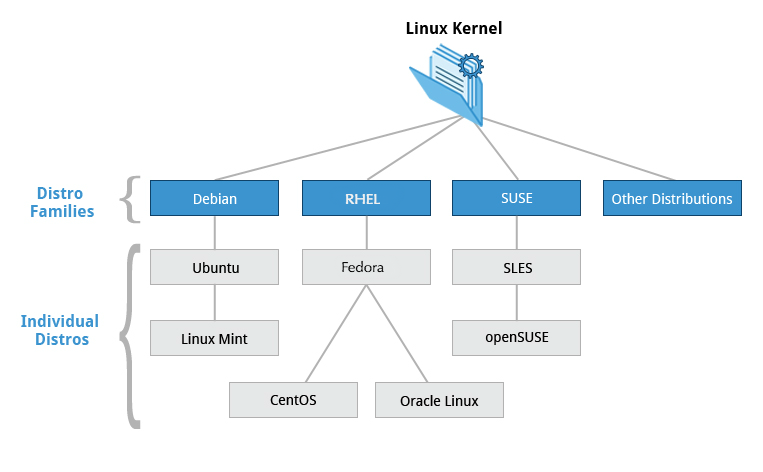
https://lwn.net/Distributions

**So, what is a Linux distribution and how does it relate to the Linux kernel?**

The Linux kernel is the core of the operating system. A full Linux distribution consists of the kernel plus a number of other software tools for file-related operations, user management, and software package management. Each of these tools provides a part of the complete system. Each tool is often its own separate project, with its own developers working to perfect that piece of the system.

While the most recent Linux kernel (and earlier versions) can always be found in [The Linux Kernel Archives](https://www.kernel.org/), Linux distributions may be based on different kernel versions. For example, the very popular RHEL 8 distribution is based on the 4.18 kernel, which is not new but is extremely stable. Other distributions may move more quickly in adopting the latest kernel releases. It is important to note that the kernel is not an all-or-nothing proposition, for example, RHEL/CentOS has incorporated many of the more recent kernel improvements into their older versions, as have Ubuntu, OpenSUSE, SLES, etc.

Examples of other essential tools and ingredients provided by distributions include the C/C++ and Clang compilers, the gdb debugger, the core system libraries applications need to link with in order to run, the low-level interface for drawing graphics on the screen, as well as the higher-level desktop environment, and the system for installing and updating the various components, including the kernel itself. And all distributions come with a rather complete suite of applications already installed.



Red Hat Family Systems (including Fedora -> CentOS, Linux Mint)

SUSE Family Systems (including openSUSE)

Debian Family Systems (including Ubuntu and Linux Mint).

**Some of the key facts about the Red Hat distribution family are:**

1. Fedora serves as an upstream testing platform for RHEL.
2. CentOS is a close clone of RHEL, while Oracle Linux is mostly a copy with some changes (in fact, CentOS has been part of Red Hat since 2014).
3. A heavily patched version 3.10 kernel is used in RHEL/CentOS 7, while version 4.18 is used in RHEL/CentOS 8.
4. It supports hardware platforms such as Intel x86, Arm, Itanium, PowerPC, and IBM System z.
5. It uses the yum and dnf RPM-based yum package managers (covered in detail later) to install, update, and remove packages in the system.
6. RHEL is widely used by enterprises which host their own systems.

**Some of the key facts about the SUSE family are listed below:**

1. SUSE Linux Enterprise Server (SLES) is upstream for openSUSE.
2. Kernel version 4.12 is used in openSUSE Leap 15.
3. It uses the RPM-based zypper package manager (we cover it in detail later) to install, update, and remove packages in the system.
4. It includes the YaST (Yet Another Setup Tool) application for system administration purposes.
5. SLES is widely used in retail and many other sectors.

**Some key facts about the Debian family are listed below:**

1. The Debian family is upstream for Ubuntu, and Ubuntu is upstream for Linux Mint and others.
2. Kernel version 5.8 is used in Ubuntu 20.04 LTS.
3. It uses the DPKG-based APT package manager (using apt, apt-get, apt-cache, etc., which we cover in detail later) to install, update, and remove packages in the system.
4. Ubuntu has been widely used for cloud deployments.
5. While Ubuntu is built on top of Debian and is GNOME-based under the hood, it differs visually from the interface on standard Debian, as well as other distributions.

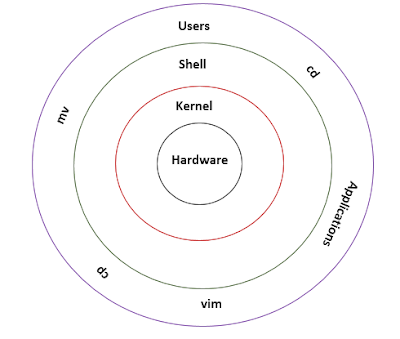
**9. Installation of Linux**

Install Linux on AWS.

**10. Linux Terminology**

When you start exploring Linux, you will soon come across some terms which may be unfamiliar, such as Kernel**, Distribution, Boot loader, Service, Filesystem, X window system, Desktop environment, Command-line, etc.**

**11. Linux architecture diagram:**



**Hardware layer**

Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc)

**Kernel**

It is the core of Operating System, interacts directly with hardware,

It controls computer hardware.

On hardware we installed one utility is called Kernal,

Kernal means we are installing Operating System.

**Shell**

An interface to kernel, convert users’ information to kernel standard format.

Once we install kernal (Operating System), It means by default you will get bash (shell).

Shell is the protective layer; SHELL will communicate with kernal.

Shell is mediator between user and kernal.

Types of Shells in Linux

1. bash - Bourne-Again shell

**Bash**is a command processor that typically runs in a text window, where the user types commands that cause actions. Bash can also read and execute commands from a file, called a script. Like all Unix shells, when a user presses the tab key within an interactive command-shell, Bash automatically uses command line completion to match partly typed program names, filenames and variable names. The Bash command-line completion system is very flexible and customizable,  
  
2. sh - Bourne shell

**The Bourne shell (sh)** is a shell, or command-line interpreter, for computer operating systems, Developed by Stephen Bourne at Bell Labs, It offers command line editing. It is easy to move cursor around and easy to edit text at command line.  
  
3. csh - c shell

**C shell**is a Unix shell created by Bill Joy while he was a graduate student at University of California, Berkeley in the late 1970s. It was designed to allow users to write shell script programs using a syntax very similar to that of the C programming language. The C shell is a command processor typically run in a text window, allowing the user to type commands. The C shell can also read commands from a file, called a script.   
4. tcsh

**TCSH** is a Unix shell based on and compatible with the C shell (csh). It is essentially the C shell with programmable command-line completion, command-line editing, and a few other features. Unlike the other common shells, functions cannot be defined in a tcsh script and the user must use aliases instead (as in csh).  
  
5. ksh - KornShell

**KornShell (ksh)** is a Unix shell which was developed by David Korn at Bell Labs in the early 1980s and announced at USENIX on July 14, 1983.  Ksh improves on the Bourne shell by adding floating-point arithmetic, job control, command aliasing and command completion. KornShell is backward-compatible with the Bourne shell and includes many features of the C shell.  
  
**Diff between windows and Unix based OS.**

**Windows:**

Virus, downtime, no sourcode, Licensed, Required 10 gb, 1gb ram.

**Unix:**

No virus, No downtime, Open source, Freeware, 256mb, 1.44 mb

**Linux:**

**REDHAT => DATABASE**

**CENTOS => DEVELOPMENT**

**UBUNTU =>TESTING OR CLIENT**

**FEDORA**

**PC-QUEST**

**REDHAT**

**----------**

**RHEL AND CENTOS**

**S/W H/W**

**SUN SOLARIS SPARC**

**HP HP-UX HP**

**IBM AIX IBM**

**These are hardware dependent. ALL Windows OS are hardware independent.**

**Windows -> file | folder | directory**

**Linux -> file| file | file => everything is file in Linux.**

**Architecture:**

**Hardware->Kernal->Shell->Applications**

**On shell we write commands: - kernel is written in c language.**

**Basic Commands Navigation and Directory control commands**

* 1. **Spinup linux machine**
  2. **Understand root and ec2 users**
  3. **Understand how to switch between users**
  4. **How to create directory**
  5. **How to delete/remove empty directory**
  6. **How to delete/remove non-empty directory**
  7. **How to create files**
  8. **How to delete files**
  9. **How to add content to files**
  10. **How to make file content zero bytes**
  11. **How to create multiple files at a time**
  12. **How to delete multiple files at a time.**
  13. **How to COPY files from one place and another place**
  14. **How to move files form one place to another place**
  15. **Difference COPY and MOVE**
  16. **Find command to search files and directories.**

1. **Check what bash/shell you are working?**

**Type ps(process)**

**Bash**

**Ps**

**# Your bash is running now, start typing commands now from here.**

**$ exit => This can kill the shell**

**$ ps => Now shell is created with different id, means new shell is created**

1. **Check which user you are logged in?**

**$ -> Normal user**

**# -> Root user**

**[Ec2user@ip ~] $**

**[root@ip~] #**

**Type exit to kill shell and connection will be closed.**

**Again, reconnect and check ps, everytime it gives new process id.**

**[Ec2user@ip ~] $ pwd**

/home/ec2-user

**// Now switch to root user**

[Ec2user@ip ~] $ **sudo su -**

[root@ip~] # **pwd**

/root => **the root user located in root directory.**

**// Now check the process whether same or not**

**# ps**

Sudo

Su

Bash

Ps

// to come out from the user root

**# exit**

**$**

**Use cd to switch between directories**

**$ pwd**

**/home/ec2-user**

**$ cd / => means go to the top most root directory**

[Ec2user@ip /] $ **ls**

Bin dev home usr proc run srv tmp var

Etc data boot media opt root sbin sys

**$ cd home**

**$ ls**

Ec2-user

**$ cd ec2-user**

**$** pwd

**/home/ec2-user**

Type **cd** from anywhere you can directly go to users working directory.

**$ cd**

**Cd =>** change directory

**Cd ..** => change the directory backward

**Cd ../../ =>** to change two directories back

**Cd (enter) =>** to go to the user home directory.

**Cd / =>** to go to the /directory (root)

**echo $HOME**

**pwd =>** present working directory

**cd ~ or cd =>** Change to your home directory

**cd .. =>** change to one level back.

**cd - =>** change to previous directory

C:\\ in windows and / (root) in Linux same.

**/ =>** root directory

note: **Root directory is different** than **root user**.

**/ root =>** is the root user home directory

**ubuntu@ip ~] $** => It is in home directory of **ubuntu user ($ => Normal user)**

**root@ip~] # =>** It is a home directory of **root user (# => Root user)**

**sudo su** => Takes you to root directory /

**sudo su -** => Takes you to **root user** home directory /root

**cd ~ =**> takes you to the user home directory

**cd –** or cd => takes you to the previous state of the working directory

**cd / =>** takes you to the root directory

We have seen

**pwd**

**ls**

**cd** commands so far.

**Working with directories**

1. **Create a Directory**
2. **Delete Directory**
3. **Create sub-Directories**
4. **Moving, Removing and Renaming directory.**

**mkdir**

**mkdir <options> <directory name>**

**The options to create a directory:**

**1. Relative Path or Relative mode**

**2. Absolute Path or Absolute mode.**

**mkdir <options> <directory name>**

**mkdir** -p dir1/dir2/dir3/dir4/dir5

**-p**  is parent

**mkdir -p dir3/ {dir1,dir2,dir3, dir4,dir5}**

**To create multiple directories with single line.**

**mkdir d{1.. 100}**

A lot of examples to create relatively and absolutely.

**Relative Path:**

$Cd /

**$ mkdir d1**

**$ mkdir d2**

**$ mkdir d3**

ls

**$ rmdir d3**

**$ rmdir d2**

**$ rmdir d1**

**Absolute path:**

$ mkdir  **-p /d1/d2/d3**

$ mkdir  **-p dir1/dir {1,2,3,4,5}**

$ rmdir **/d1/d2/d3**

**How to create hidden file or directory**

Ans: mkdir . python -> use. to create hidden directory

**How to see the files using absolute path?**

**ls /d1/d2/d3**

**How to display Directories/Files in Order?**

**ls -lt** -> Going to display in descending order.

**ls -ltr** -> r reverse order t-

**What is the root user home directory?**

**Ans: /root**

**Welcome**

**$ mkdir dir3 && cd dir3 && touch file4 && echo welcome**

&& => you can append more commands

**echo $?** => 0 or 1 => 0 is success 1 is failure

Its internal work while appending commands.

dealing with lengthy commands: -

**$ mkdir dir5 && cd dir5 && cat file1**

welcome.

**remove**

**rmdir**

The Linux rmdir command removes **empty** directories only.

The command uses the following syntax:

Syntax: -

**Rmdir [options] [directory name]**

The **rmdir** command includes the following options:

1. --ignore-fail-on-non-empty => Doesn't show an error message when trying to remove a non- empty directory.
2. -p => Removes the directory along with its parent in the hierarchy.
3. -v => Provides a verbose output.
4. –help=> Displays help text.
5. –version => Displays the command version.

To remove **non-empty** directory we use **rm** command options include:

1. -f => Forces the removal of all files or directories.
2. -I => Prompts for confirmation before removing.
3. -I => Prompts once before removing more than three files or when removing recursively.
4. -r => Removes directories and their content recursively
5. -d => Removes empty directories.
6. -v => Provides a verbose output.
7. –help =>Displays the help text.
8. –version => Displays the command version.

**Note: Without checking** *pwd* **don’t use this command, it’s dangerous**

**Listing Files or Directories**

**Ls => listing of files and directories.**

**Ls - l => long listing**

**Ls -p =>**

**Ls -s => to see the size of the files, default in bytes.**

**Ls -a => list of all hidden files**

**Ls -i => to see the files with inode numbers**

**Ls -l <file name> => to see the file information**

**Ls -ld <directory name> => To see the directory information.**

**Ls -m => mode**

**Ls -vp =>**

**Ls -v =>verbose**

**Ls -lt => going to display in descending order.**

**Ls =ltr => reverse order.**

**Ls -li => it displays with inode numbers**

**Ls -l => listing along with size but not in human readable format.**

**Ls -lh => listing along with size.**

**Working with Files without any editor**

**Syntax of command:**

**Command Options Arguments**

**To create files:**

1. **echo**
2. **touch**
3. **cat**
4. **File file1 (ASCII text file)**

Sometimes, you may want to create a short file and don't want to bother invoking a full text editor. If you want to create a file without using an editor, there are two standard ways to create one from the command line and fill it with content.

**The first** **way** is to use echo repeatedly:  
  
**$ echo line one > myfile  
$ echo line two >> myfile  
$ echo line three >> myfile**

**Note**: single greater-than sign (>) will send the output of a command to a file two of them (>>) will append the new output to an existing file.

**echo**

echo simply displays (echoes) text. It is used simply, as in:

**$ echo string**

echo can be used to display a string on standard output (i.e., the terminal) or to place in a new file (using the > operator) or append to an already existing file (using the >> operator).

The –e option, along with the following switches, is used to enable special character sequences, such as the newline character or horizontal tab:

* \n represents newline
* \t represents horizontal tab.

echo is particularly useful for viewing the values of environment variables (built-in shell variables). For example, echo $USERNAME will print the name of the user who has logged into the current terminal.

The following table lists echo commands and their usage:

**echo string > newfile** => The specified string is placed in a new file

**echo string >> existingfile** => The specified string is appended to the end of an already existing file

**echo $variable** => The contents of the specified environment variable are displayed

1. The **Second** way is to use cat combined with redirection:

**cat** is short for concatenate and is one of the most frequently used Linux command line utilities. It is often used to read and print files, as well as for simply viewing file contents. To view a file, use the following command:

**$ cat <filename>**

For example, **cat readme.txt** will display the contents of **readme.txt** on the terminal. However, the main purpose of cat is often to combine (concatenate) multiple files together. You can perform the actions listed in the table using cat.

The **tac** command (cat spelled backwards) prints the lines of a file in reverse order. Each line remains the same, but the order of lines is inverted. The syntax of tac is exactly the same as for cat, as in:

**$ tac file  
$ tac file1 file2 > newfile**

**$ cat file1 file2** => Concatenate multiple files and display the output; i.e. the entire content of the first file is followed by that of the second file

**$ cat file1 file2 > newfile** => Combine multiple files and save the output into a new file

**$ cat file >> existingfile** => Append a file to the end of an existing file

**$ cat > file** => Any subsequent lines typed will go into the file, until CTRL-D is typed

**$ cat >> file** => Any subsequent lines are appended to the file, until CTRL-D is typed

**Using cat Interactively**

**cat** can be used to read from standard input (such as the terminal window) if no files are specified. You can use the > operator to create and add lines into a new file, and the >> operator to append lines (or files) to an existing file. We mentioned this when talking about how to create files without an editor.

To create a new file, at the command prompt type cat > <filename> and press the Enter key.

This command creates a new file and waits for the user to edit/enter the text. After you finish typing the required text, press CTRL-D at the beginning of the next line to save and exit the editing.

Another way to create a file at the terminal is cat > <filename> << EOF. A new file is created and you can type the required input. To exit, enter EOF at the beginning of a line.

Note that EOF is case sensitive. One can also use another word, such as STOP.

**$ cat << EOF > myfile  
> line one  
> line two  
> line three  
> EOF  
$**

Both techniques produce a file with the following lines in it:

line one  
line two  
line three

1. Create a new or empty file

2. View contents of a file

3. Changing the timestamp of a file

4. Moving Removing or Renaming a file

**To make file empty:**

**0 > file1 => now file1 is empty.**

**Touch:**

**1. To create empty files**

**2. To create multiple files with zero bytes.**

**3. To change the timestamp of file**

**4. To create the file with timestamp.**

**To change time stamp on file:**

**Touch file1 => it will change the file time from old to new.**

**$ touch** sample file for devops training

**// here it creates 4 files.**

**$ touch** “sample file for devops”

**$ rm f1 => say yes or no**

**$ rm -f f2 => -f means forcible**

**$ rm f1 f2 f3**

**$ rm \*. => to remove all files. which are interactive asks yes or no**

**$ rm -f \* => to remove all files**

**rm removes only files not directories or its subdirectories.**

**$ rm => to remove files.**

**$ rm -f /dir1/\***

**$ rm /dir/\***

**$ touch f1 f2**

**$ ls**

**$ cat f1 f2**

**$ cat > f1** (asking for input data)

First line

Second line

Third line

Fourth line

Ctrl +d

\*\*\*\*\*\*

To display the file data

**$ cat f1**

//content is displayed

\*\*\*\*\*\*\*

**$ cat -n f1**

// display data with line number

**\*\*\***

**$ cat > f1**

Fifth line

Sixth line

Seventh line

\*\*\*\*\*\*

**$ cat > f1**

//previous content lost

Fifth line

Sixth line

Seventh line

\*\*\*\*\*

**// Just append the content**

**$ cat >> f1**

Eight lines

Nine lines

\*\*\*\*

**$ cat f1**

Fifth line

Sixth line

Seventh line

Eight line

Nine line

\*\*\*\*\*

**$ tac f1**

**// reverse print usning tac**

Nine line

Eight line

Seventh line

Sixth line

Fifth line

\*\*\*\*\*\*

**$ cat > f1** // this displays the content

Fifth line

Sixth line

Seventh line

Eight line

Nine line

// Instead of printing, contents should copy to the next line.

**$ cat f1 > f6** // f1 content is copied to f6

Cat f6

Fifth line

Sixth line

Seventh line

Eight lines

Nine lines

\*\*\*\*\*\*\*\*

**$ cat f6 > f9**

**$ cat f9**

Fifth line

Sixth line

Seventh line

Eight lines

Nine lines

\*\*\*\*\*\*\*\*

**$ cat f6 >> f9** //it will append the data

**$ cat f9**

Fifth line

Sixth line

Seventh line

Eight lines

Nine line

Fifth line

Sixth line

Seventh line

Eight line

Nine line

**$ echo command.**

**$ echo welcome** // prints as it is

Welcome

Echo first line

first line

\*\*\*\*\*\*\*\*

**$ echo first line > f2**

**$ echo second line >> f2**

**$ echo third line >> f2**

\*\*\*\*\*

**$ cat f2**

First line

Second line

Third line

**Tree**

Bird's eye view of file system

tree

tree -d

**Limitations of cat**

1. **Cannot edit or delete text**
2. **Only creation and appending**

**Navigating the directory history**

**pushd**

**popd**

**dirs**

**tac**

**tail**

**head**

**less**

**What is the diff between cp and mv?**

Navigating the directory history

pushd

popd

dirs

**COPY:**

**Syntax:**

**Cp <options> <source> <destination>**

**Moving Removing or Renaming a file or directory**

**Mkdir dir1/**

**Cd /dir1**

**Touch f {1….100}**

**Mkdir /dir2**

**Cp f1 , f2, f3 /dir2**

**Cp \* /dir2 => source relative and destination absolute.**

**Cd /dir2**

**Cp /dir1/\* . => current directory i.e, dir2**

**Cp /dir1/\* /dir2**

**Note:** cp copies only files not directories, and subdirectories.

**Mkdir d {1.. 100}**

**Cp /dir/\* /dir2**

**Omitting directories**

**cp -r /dir/\* /dir2**

**copies all directories and files. (Non interactive mode)**

**cp -rv /dir1/ \* /dir2**

**v=> verbose.**

**MOVE:**

Moving the files from one dir to another dir

**Cd /dir1**

**Mv f1 file1 => rename**

**Mv fi /dir1 => moving**

**Mv f3 /dir2/dir3 => move and rename.**

**Mkdir /dir1**

**Cd /dir1**

**To remove file directory and its subdirectories.**

**Mkdir d1 d2 d3**

**Rm -f \***

**Rm -rf \* => removes files/directories and subdirectories**

**Rm -rf /dir1/\* -> removes all files in dir1\**

**Rm -rf /dir1 => removes dir1**

**Rm -rf / => to remove all from root directory.**

**CP and MV**

**Copy from one to another**

**$ Cp f1 dir2 => copy from source to destination.**

**Destination should be directory**

**$ cp f1 f2 => this will delete the contents of f2 and paste f1**

**Copy directory to directory**

**$ cp -r dir8 dir7**

**-r => recursively**

**Cut and paste**

**$ mv dir4 dir5**

**$ls dir5**

**$ mv sample/ dir4 => creates the dir4 and move the sample fiel.**

**Rename:**

**$ mv file1 sampellinuxfortest**

**$ls**

**Sampellinuxfortest**

**$ mv Sampellinuxfortest file1**

**$ ls**

**File1**

**cat**

**delete multiple files**

$ rm file {1..100}

$ rm fi\*

$ rm -r d\* => remove directories start with d.

echo => to print some content on prompt

$ echo hello; echo world; => ; is a new line

$ echo hello && echo world;

**$ mkdir dir3 && cd dir3 && touch file7 && echo welcome file7 && cat file7**

**$ cat file7**

**// special variables**

**// if the previous command executed successfully, it returns 0**

**$ echo $?**

**0**

**// if the previous command executed successfully, it returns 1**

**$ echo $?**

**1**

**$ mkdir fdir5 \**

**&& cd dir5 \**

**&& touch file8 \**

**&& echo welcome >> file8**

**&& cat file8**

**$ vi test.sh**

**// write all commands here as script file**

**$ bash test.sh**

**$ extension is required to identify the files.**

**Find command**

Find is used to search and locate list of files and directories based on conditions you specify for files that match the arguments. Find can be used in variety of conditions like you can find files by permissions, users, groups file type, date size and other possible criteria.

How to Search all the empty files in the current directory?

**Ans) find. -type f -empty**

How to search al the empty files in the user home directory?

**Ans) find ~ -type f -empty**

Search files in Entire server?

# **find / -type f -empty**

**#find /opt/ -type f -empty**

**How to search all the directories in the current directory?**

**Ans)** **$ find . -type d -empty**

**How to search al the empty files in the user home directory?**

**Ans) $** **find ~ -type d -empty**

**How to Search directories in Entire server?**

**# find / -type d -empty**

**How to search directories in /opt directory?**

**#find /opt/ -type d -empty**

**# find -name abc.sh** => It will search the abc.sh in the current directory.

**# find -iname abc.sh** => It will search the abc.sh file under current directory ignoring the case.

**# find . -type f -perm 0777** => It will search all the files which have 777 permissions.

**# find . -perm /a=x** => It will search all the files which have execute permissions.

**# find / -perm 600** => it will display the files and directories which have the 600 permissions,

-perm option is used to find files based upon permissions.

**# find . -name “\*.java” | xargs grep “FTPReadXML”**: => It will display java files which have the FTPReadXML word in those files.

**# find . -name “\*.java” -exec chmod 700 ‘{}’ \;** :=> It will set access permissions (700) for all java files.

The argument ‘{}’ inserts each found file into the chmod command line. The \; argument indicates the exec command line has ended.

**# Find -size +1000c -exec is -l {} \**: It will display the files which size greater than 1000 bytes.

**# find /logs/ -mmin +30 -exec rm -f {} \;** => It will remove the logs which are created more than 30 minutes back in /logs directory.

**# find . -name “\*” -ls | awk ‘{print $7 “” $11}’ | sort -rn |head -n 10** => Find the largest files in the current directory and recursion through all subdirectories.

**The files related to irrespective of the user and the shell:**

**/etc/motd**

**/etc/profile**

**/etc/bashrc**

These files also called global profiles or global environment files.

**/etc/motd** => is the file form message of the day. The message in /etc/motd file execute on user console when user logged in. These messages only for new login but not for logged in users.

Vi /etc/motd

Good evening

Wq!

To see the welcome message:

Press ctrl+alt+F2

Ctrl+alt+F1 to see desktop version.

**/etc/profile** => The messages in /etc/profile is for all users. only new login users not logged in users.

Vi /etc/profile

Press esc+shift+g => to go to last line in page and last line

Echo welcome to India

To see messages ctrl+alt+F2

**/etc/bashrc** => the messages in /etc/bashrc file is only for new logins

Press esc shift g => to go to last page and last line.

Echo: have a nice day.

To change the umask value permanently or update /etc/bashrc file.

The default umask value for normal user is 002

The default umask value for root(super) user 022

**The file related with respective of users and the shell:**

.bash\_profile => execute when user logged in.

.bash\_logout => execute when user logged out.

.bash\_bashrc => for bash shell.

.bash\_history => maintains the history of all commands.

These files are also called user environment files or user profiles resides in every user home directory.

Etc/bashrc => global

.bashrc => local

The user default parameters are stored in /etc/default /useradd

# vi /etc/default/useradd

HOME = /home

INACTIVE = -1

EXPIRE =

SHELL = /BIN/BASH

SKEL = /ETC/SKEL

/etc/skel => is the directory contains the user environment files or user profiles the files under /etc/skel copied to the user home directory when the user is created with useradd command.

/etc/login.defs file contains the user default login values.

**Commands worked so far:**

cat

tac

tail

head

less

touch

mkdir

rmdir

mv (move and rename a file)

rm

rm -f

rm -i

rm -rf

**mkdir** with **vpm** options are important.

**USER ADMINISTRATION**

1. **Create user - useradd**
2. **Set password - passwd**
3. **Delete user -**
4. **Update user**
5. **Create group**
6. **Delete group**
7. **Update group**
8. **Add single user to group**
9. **Add multiple users to group**
10. **How to make user as admin**
11. **How to create multiple admins in a group**
12. **password less authentication to access the instances. (without. pem)**
13. **Lock user**
14. **Unlock user**
15. **Lock group**
16. **Unlock group**
17. **Find where users stored and passwords to users stored**
18. **Find where group stored and passwords stored.**

**Types of users in Linux OS?**

1. **Super user or admin or root user => at the time of installing OS will create.**
2. **Normal user => ec2-user, vinodh, bhanu, teja**
3. **System user =>** suppose if you install Jenkins server it will create Jenkins user same privileges as of normal user, Apache HTTP server creates an apache user which is again an apache user.**ftp, ssh, apache2, jenkins**

Three types of users

**Super User** - It's username is root. root can access all directories in an os. It's UID is zero (0)  
  
**Normal Users** - Normal users can access only limited directories. It's UID range is 500 - 60000  
  
**System Users** - Users created automatically at the time of Operating system installation or at the time of software installation. It will be a non-login users. It's UID range is 1 - 499

**Once normal user is created, he will automatically get**

**UID,**

**GID,**

**MAIL BOX,**

**HOME DIRECTORY,**

**SHELL,**

**DESIGNATION.**

**(Like bank account opening, atmcard, checkbook, online account)**

**Find where users and passwords are stored.**

**Find where group stored and passwords stored.**

**Answer:**

**/etc/passed => you can see all usernames here**

**/etc/shadow => you can see all passwords here**

**/etc/group =>you can see all groups here**

**/etc/gshadow => you can see all group passwords here**

**# cat /etc/passwd**

**Root: X: 0:0: root:/root:/bin/bash**

**ec2-user: 1001:1001: :/home/ec2-user: /bin/bash**

**Root => user name**

**Vinodh => normal user**

**X=> link to the password file**

**0: => userid**

**0=> groupid**

**Root => designation**

**/root => home directory location**

**/bin/bash => shell**

1. **Create user**
2. **Update the password**
3. **Update the designation**
4. **Convert normal user into root user**
5. **Sudoers file**
6. **Delete user**
7. **Update user**
8. **Groupadd**
9. **Gpasswd**
10. **chage**

**Create User**

**Useradd -u <uid> -g <gid> -G <gid> -c<comment> -d /home/<homedir> -m -s /bin/<shell> <usernaeme>**

**g -** primary group

**G-** secondary group

**c-** type of user

**d –** path of user home directory

**m-** make directory

**s-** shell (shell command interpreter)

**$ useradd vinodh**

**Permission denied. // Normal user have no right to create users.**

**$sudo su -**

**# Whoami**

**Root**

**# useradd sorties**

**In the below file you can see all user’s information.**

**# cat /etc/passwd**

**Sorties: 1001:1001: /home/sorties:/bin/bash**

**Sorties => username**

**X => link to the password**

**# cat /etc/shadow**

**sorties: !!: 18987:0:99999: 7: : :**

**!! => means password is not created/available.**

**How to set the password to the user**

***If you don’t give user name to which password needs to set, it will assign the password to the current user.***

**password to the current user**

**# passwd**

**Changing password for user root.**

**New password:**

**# passwd sorties**

**Changing the password to user vinodh**

**New password: XXXX**

**# useradd ramu**

**# passwd ramu**

**Now 2 users are created, you can see these at**

**# cat /etc/passwd**

**vinodh:x: 1001:1001: :/home/vinodh: /bin/bash**

**ramu:x: 1002:1002: :/home/vinodh: /bin/bash**

**/bin/bash => shell is attached to the user automatically**

**If you want to update anything like designation you can use usermod.**

**// need some work here \*\*\*\*\*\***

**# Usermod**

**// u can see lot of options here**

**# Usermod –help**

**// here with help command lot of options for usermod you can see ..**

**// id display the current user information with id**

**# id // it will display current user id.**

**# id sorties // display user vinodh information**

**Uid=1001(sorties) gid = 1001(sorties) groups = 1001(sorties)**

**# id ramu**

**# su – ramu**

**# Whoami**

**Ramu**

**$ pwd**

**/home/ramu**

**// now suppose we asked ramu to install some package.**

**//using ramu user let us install some commands**

**$ yum install vim -y**

**Error: This command needs super user privileges**

**$ sudo yum install vim -y**

**Ramu $ sudo useradd Manoj**

**Error: permission denied.**

**The user ramu is not in sudoers file.**

**$ exit**

**In the above commands you are trying to add the manoj user from ramu user using sudo command,**

**Eventhough ramu user is using sudo, this ramu user needs to be added in one file called /etc/sudoers**

**Now the administrator will add this ramu user to sudoers file**

**The sudoers file location is /etc/sudoers**

**All administrators are familiar with this file.**

**How to open sudoers file?**

**# visudo**

**// explain the sudoers file.**

**// we can restrict here ramu to use all commands or some commands**

**Ramu ALL=(ALL)**

**[root]:**

**Sudoers**

**Ec2-user ALL=(ALL) NOPASSWD: ALL**

**Ramu ALL=(ALL) NOPASSWD: ALL**

**// what is this NOPASSWD: ALL means**

**now while installing any software or creating user it won’t ask for password.**

**$ On behalf of root user, we are running the commands.**

**// now you can add any commands here**

**$ sudo useradd r1**

**All the commands are recorded by root.**

**GROUP:**

**// Now let us try to understand the group**

**// first let us find the information in the group file.**

**// open the /etc/group**

**[#] cat /etc/group**

**// lot of groups are there, from 1000, user groups created/started.**

**// these are all called normal groups**

**Ec2-user:X:1000:**

**Sorties:X:1001**

**Ramu:X:1002**

**R1:x:1003**

**# sudo yum remove vim -y**

**// now how to create group**

**// create group now**

**# groupadd devops // here in the group we can add users**

**// now open the gshadow file**

**// you can see all groups are there.**

**# /etc/gshadow**

**# cat /etc/gshadow**

**Sorties: ! : ::**

**R1 : ! : :**

**devops: ! : :**

**// now add people to the devops group using usermod**

**# usermod -G devops sorties**

**# usermod -G devops ramu**

**now open gshadow file and verify**

**# cat /etc/gshadow**

**Sorties: ! : ::**

**R1 : ! : :**

**devops: ! : :sorties**

**create multiple users and add all to group.**

**# useradd u1**

**# useradd u2**

**# useradd u3**

**# useradd u4**

**Using usermod command you can add one user at a time to the group**

**# usermod -G devops u4**

**# usermod-G devops ramu**

**# usermod-G devops u3**

**To add multiple users at a time we have to use gpasswd commnad.**

**If you want to see all the commnads under gpaawd**

**# gpasswd – help // see all options**

**# gpasswd -a u1 devops. // If you want to add user to existing group you can add.**

**# gpasswd -M u1, u2, u3 devops. // This adds multiple users to devops group.**

**How to make a user as admin?**

**# gpasswd -A sorties devops**

**Can we have multiple admin in group?**

**# gpasswd – A u1 devops**

**// It will override, for one group one admin only**

**Commands we have used so far is:**

**Useradd groupadd**

**Passwd passwd**

**Visudo gpasswd**

**Usermod**

**Delete the group**

**If you delete the user, home directory and mailbox won’t be deleted**

**# groupdel devops**

**# userdel vinodh**

**# cd /home/vinodh**

**# cd /var/spool/mail**

**Mail # ls**

**u1 u2 u3 vinodh ramu**

**How to delete the user permanently?**

**This removes mail and home directory.**

**# userdel -r ramu**

**Understand this new concept**

**# useradd rajesh**

**# passwd rajesh**

**# visudo**

**Rajesh ALL = (ALL) NOPASSWD: ALL**

**$ ssh -I “key” rajesh@ip // here permission denied to rajesh user.**

**// by default, AWS is not allowed with password mechanism**

**// What are the changes I need to do rajesh user to access my aws machine.**

**There is one configuration file to handle this sshd\_config , this is system level setting not user level setting.**

**# vi /etc/ssh/sshd\_config**

**PasswordAuthentication yes.**

**// here previously no, were there that’s why you are not able to login with the password.**

# systemctl reload sshd

You have to provide a service name sshd not a configuration file, to maintain this service, you need a configuration file. whenever you modify any configuration files you need to **reload the service.**

**Now, login**

**$ ssh -I “key” rajesh@ip**

**// enter the password**

**Now password authentication is enabled, Now I am able to login with ssh**

**Whenever we pass authentication is yes enabled then no keypair is required.**

**Now try without any key-pair to login, whenever you login with ssh no keypair is required.**

**$ ssh rajesh@ip**

**Enter new password: \*\*\*\*\*\*\***

**// now you are able to connect without pem file.**

**$ exit**

**// In your company you can do like this and can access with your username only.**

**Suppose Rajesh is leaving the company, by today 4 pm.**

**How to lock the user temporarily?**

**Now rajesh is leaving the company:**

**$ sudo -i**

**# usermod -L rajesh**

**$ ssh rajesh@ ipaddress**

**Permission denied try again later.**

**# cat /etc/shadow**

**// in this file the exclamatory mark is there. Which means user is locked**

**Now I don’t want to go to shadow file, How to check the status of the password**

**# passwd -S rajesh // all rajesh data came.**

**Showing password is locked**

**How to unlock the user?**

**# passwd -U rajesh // password is unlocked**

**# passwd -S rajesh // Now who are the 3 users logged into the machine.**

**How many users logged into this particular system?**

**# who // this command displays the no of user logged in**

**ec2-user #**

**root #**

**rajesh #**

**Let us try to understand one more small concept:-**

**Create one more user?**

**# useradd suma**

**# passwd suma**

**# visudo**

**Suma ALL ALL**

**Set the age of the user?**

**# chage -d O suma // as long as suma logged in it will ask for the password.**

**No of days - -d user has to change his password**

**0 – zero => means as soon as the user logged in he has to change the password.**

**// for contract employees this will be set like this**

**How to update the password?**

**# chage -l suma** // all information displayed here, if you need to change particular user.

**File Administration**

1. **Create files using ec2-user**
2. **Create files using root user**
3. **Change the ownership of the file**
4. **Change the ownership of the group**
5. **Change the ownership of the user and group at a time**
6. **Permissions**
7. **How to add permission to file**
8. **How to remove permission to file**
9. **How to add permission to directory**
10. **How to remove permission to directory**
11. **Umask**
12. **SUID SGID STICKBIT (Need to work on it)**
13. **ACCESS CONTROL LISTS**

**ec2-user$ touch file1 file2 file3**

**$ ls -l**

**These files belong to the ec2-user group and ec2-user is the owner.**

**Create files using root user**

**# touch file1 file2 file3**

**# ls -l**

**These files belongs to the root group and root is the owner.**

**# touch sample**

**This file root is the owner .**

**Change the ownership of the file**

**# useradd bhanu**

**# groupadd devops**

**# chown bhanu sample**

**# ls -l**

**Bhanu is owner now, root is the group ,**

**# chgrp devops sample**

**# ls -l**

**Bhanu is owner now, devops is group.**

**We can change both at a time.**

**# touch file100**

**# chown bhanu:devops file100**

**# ls -l**

**# mkdir dir1**

**# ll**

**Root root**

**# chown bhanu:devops dir1**

**# ll**

**Bhanu devops**

**If any error occurs, do it recursively using -R**

**# chown -R bhanu:devops dir1**

**When more files are there how can you change the group and users?**

**Permissions:**

**Whether it is file or directory these permissions are same**

0 =>No permission

1 =>execute permission

2 =>write permission

4 =>read permission

5 =>read and execute permission

6 =>read and write permission

7 =>read write and execute permission

**r- read - 4**

**w- write - 2**

**x – execute - 1**

**7 – r w x**

**6 – r w -**

**5 – r - x**

**The order to provide permissions is:**

**U G O**

rwx rwx rwx

user group other

**rwxr- -r- - => meaning of this is user can read write and execute group can read others can read .**

**Whenever you create a file or directory you will get some default permissions.**

**# l l**

**# chmod u+x, g+wx, o+rx file1**

**# l l**

**// now file1 got additional permissions**

**You can remove permissions as well**

**# chmod o – wx file1**

**// now file1 permissions removed**

**# ll**

**# ls -l file1**

**#**

**# Touch <file1> to create a file**

**# Touch file1**

**# Ls -l <filename> => to see the file permissions**

**# mkdir dir1**

**# ls -ld dir1 => to see the directory permissions.**

**UMASK Command**

The term **umask** refers to two things:

1. The Linux umask command. umask (user file-creation mode) is a Linux command that lets you set up default permissions for newly created files and folders.

2. A user-defined permissions ‘mask’. A user can choose how to restrict permissions by using a **permissions mask**. A permission mask interacts with the default system permissions and changes them. The umask command is used to apply this mask.

**How Does Umask Work?**

The umask command works by affecting the default [Linux file and folder permissions](https://phoenixnap.com/kb/linux-file-permissions).

There are three categories of permissions for every file and folder in Linux:

* **User:** Defines permissions for each individual user. By default, the user who creates a file or folder is set as the owner.
* **Group:** Defines permissions for a group of users that share the same level of access.
* **Other:** Defines permissions for anyone not included in the two previous categories.

**Umask/default file Directory**

Umask for root user 0022 0644 0755

Umask for normal user 0002 0664 0775

**How to change the default umask values?**

Base Permissions for a file 0666

Base Permissions for a directory 0777

**0666 0777**

**-0022 -0022**

**--------------------------**

0644 0755 => default values for file and directory.

1. **rwxrwxrwx what is the numeric to this?**
2. **777**
3. **R- -r - - r - - what is the number to this?**
4. **444**

**Understand how to change permissions in numeric way?**

**# mkdir dir3**

**# ls -ld dir3**

**# umask // to check the umask value.**

**0022**

**# chmod 444 file1**

**# ls -l file1**

**# chmod 777 file1**

**# ls -l file1**

**How to change the default file and directory permissions?**

**Ans: chmod**

**How to set umask value permanently?**

**Ans: $** vi ~/. Bash\_profile

Umask 0222

Setting here is applicable to this user only

**If you need to change umask value permanently you need to change in**

**#**Vi etc/profile

Umask 0222

**what are the default permissions assigned to a file or directory?**

for file: 644 (root user),664(normal user)

for directory: 755(root user), 775(normal user)

**Difference between sudo su – and sudo su?**

**Sudo su –** => It will switch to root user and it will point to the root directory, Loads root user configurations

**Sudo su** => It will switch to root user and it will not point to the root directory, it will not load root user configurations

**chmod chown chgrp**

**chmod =>** To change the file or directory permissions (rwx)

**Chown =>** To change the ownership of the file or directory Only root user can execute this command.

**Chgrp =>** To change the group ownership of a file or directory. Only root user can execute

This command.

**How to remove permissions to a file?**

**Ans: chmod 000 devops.txt**

mkdir -m 700 devops.txt

chmod ugo+r Devops.txt

the above 2 lines are the same**.**

Chmod ugo+x Devops.txt

Chmod ugo-x Devops.txt

**There are 3 types of users on file and directories**

1. user 2. Group 3. Others

U G O

1. **inode**
2. **umask**

Ls -li i => inode number

Umask => to see the current umask value.

Umask 024 => to change the umask value.

Ls -l file1

**What should be the umask value to create file or directory with original permissions.**

Umask 000

Mkdir /dir1

Cd /dir1

Touch f1 f2 f3

Ls – ld /dir1

**Useradd student** => to create the user student

Su -student => to login to the user student.

Touch a1 a2 => permission denied because no write permission to user student.

**Chmod:**

To change the file or directory access permissions.

To give the permissions in symbolic value.

Chmod ugo+rwx <file or dir>

To give the permission in octal value.

Chmod 777 <filename or directory name>

Chmod o+w /dir1

su -student

cd /dir1

Touch a1 a2 a3

Ls

a1 a2 a3

**Do not allow the user student to each other into dir1**

Chmod o -x => remove the execute permission

Su -student

Cd /dir1 => permission denied

Chmod 777 /dir1 => the permissions applied to only dir1 and subdirectories and files.

Chmod -R 777 /dir1 => the permissions applied to all the directories and files under dir1.

**Chgrp:** To change the permissions to group.

Groupadd dba => create a group dba.

Mkdir /d1

Ls -ld /d1

For directory /d1 give the write permissions for all users under dba group.

Chgrp dba/d1

Ls -ld/d1

Chmod g+w /d1.

To create the users with dba group

Useradd -g dba u1

Useradd -g dba u2

Su -u1

Cd /dir1

Touch a s d f

To change the group dba for /d1 including all files and subdirectories

**Chgrp** -R dba /d1

**Chown => to change the ownership permissions.**

Give the write permission on file1 for user student.

Chown student file1

Ls -l file1

Su -student

Cat >> file1

This is my second line of fiel1

Ctrl + D

**Group Administration**

**Group:** A group is a collection of users who can share the files and the other system resources. A group consists of group name, and the list of users belong to that group.

These groups classified into 2 types:

1. Primary group

2. Secondary group

**Primary Group:** It’s a group that every user must belong to, without a primary group the user cannot created, and the user cannot have more than one primary group.

**Secondary Group:** It’s a group the user can also belongs, the user can belong to max 15 groups.

**Syntax:**

**group add -g <gid > <gname>**

0-499 => are system defined till version 6.0

500- 60000 ports are usable.

0 to 999 are system defined in ver 7.0

1000- 60000 ids are used.

**Groupadd -g 500 dba =>** Created the group with users defined group id.

**Groupadd -g dba** **=>** created the group with system defined id.

If -g <gid> is not defined system takes default id s from 1000.

**/etc/group =>** is the file contains all the group names and group ids.

**Cat /etc/group** => to see all group ids and names.

**Grep dba /dba/group** => to see the specified group called dba.

**Syntax to create User creation:**

Useradd -u <uid> -g <gid> -G <gid> -c<comment> -d /home/<homedir> -m -s /bin/<shell> <usernaeme>

**g-** primary group

**G-** secondary group

**c-** type of user

**d –** path of user home directory

**m-** make directory

**s-** shell (shell command interpreter)

**Inode**

**Inode** -> is a data structure it will store file or directory information (size and permissions)

An inode doesn’t store the content of the file and filename.

Ls -li -> it displays with inode numbers

Ls -l -> listing along with size but not in human-readable format

Ls -lh -> listing along with size.

echo $HOME

pwd -> present working directory

cd ~ or cd ->Change to your home directory

cd .. -> change to present directory

cd - -> change to previous directory.

**Before get in to linux file system we will see the Linux users and their locations.**

Linux file system is case-sensitive. Hidden files start with a (.) extension

User and group administration related commands: -

Uid

Useradd

RHEL 7.x onwards UID starts from 1000

Before that version starts from 500

**Usermod L**

**Usermod l**

**Usermod g**

**Cp:** Moving the files from one dir to another dir

**What is the diff between cp and mv?**

File => checked based on content

Wc => counts the number of lines, words, bytes, or characters in a file.

Ln => It will create the link between files.

**What is the use of link files?**

Hard link soft link

If the permissions started with l It means the file is having a soft link.

**To find what OS using:**

Uname -s

Linux

Cat /etc/issue => to find what feature of OS we are using.

Echo $ SHELL => default shell

/bin/bash => to know the current shell

Sh => to change the shell.

Sh -4.1 # csh

Echo $0 => to know the current working shell

# exit

**reation of user UPG (user private group)**

User name: student /etc/passwd

Group name: student /etc/group

Home dir: student /home dir

Mail box: student /var/spool/mail dir

**To create password for user:**

**Passwd student**

**/etc/passwd** => file contains all users information

**/etc/shadow** => file contains all users password information

**The files related to irrespective of the user and the shell:**

**/etc/motd**

**/etc/profile**

**/etc/bashrc**

These files also called global profiles or global environment files.

**/etc/motd** => is the file form message of the day. The message in /etc/motd file execute on user console when user logged in. These messages only for new login but not for logged in users.

Vi /etc/motd

Good evening

Wq!

To see the welcome message:

Press ctrl+alt+F2

Ctrl+alt+F1 to see desktop version.

**/etc/profile** => The messages in /etc/profile is for all users. only new login users not logged in users.

Vi /etc/profile

Press esc+shift+g => to go to last line in page and last line

Echo welcome to India

To see messages ctrl+alt+F2

**/etc/bashrc** => the messages in /etc/bashrc file is only for new logins

Press esc shift g => to go to last page and last line.

Echo: have a nice day.

To change the umask value permanently or update /etc/bashrc file.

The default umask value for normal user is 002

The default umask value for root(super) user 022

**The file related with respective of users and the shell:**

.bash\_profile => execute when user logged in.

.bash\_logout => execute when user logged out.

.bash\_bashrc => for bash shell.

.bash\_history => maintains the history of all commands.

These files are also called user environment files or user profiles resides in every user home directory.

Etc/bashrc => global

.bashrc => local

The user default parameters are stored in /etc/default /useradd

# vi /etc/default/useradd

HOME = /home

INACTIVE = -1

EXPIRE =

SHELL = /BIN/BASH

SKEL = /ETC/SKEL

/etc/skel => is the directory contains the user environment files or user profiles the files under /etc/skel copied to the user home directory when the user is created with useradd command.

/etc/login.defs file contains the user default login values.

**Ownership**

**[$] touch file file1 file2 file3 file4**

**[$] ls -l**

**[$] sudo -i**

**[#] touch sample file1 file2 file3**

**[#] ls -l**

**[#] useradd vinodh**

**# groupadd devops**

**# chown vinodh sample**

**# ls -l**

**// you can see ownership changed**

**// Now you can change the group ownership of the file.**

**# chgrp devops sample**

**# touch file100**

**# ls -l**

**# chown vinodh: devops file100**

# mkdir dir1

# ll

# chown vinodh:devops dir1

# chown -R vinodh:devops dir1

For a file or directory there is some default permissions

# mkdir /test

#ls -l

// check the permissions and owner

# groupadd devops

# chgrp devops /test

# ls -ld /test

# chmod 777 /test

# su – vinodh

[vinodh]$ cd /test

[vinodh]$ touch file1

[vinodh]$ ll

Now it is taking users primary group , its not taking devops group .

Whenever file is created in some other folder .

1. **inode**
2. **umask**

Ls -li I => inode number

Umask => to see the current umask value.

Umask 024 => to change the umask value.

Ls -l file1

**What should be the umask value to create file or directory with original permissions.**

Umask 000

Mkdir /dir1

Cd /dir1

Touch f1 f2 f3

Ls – ld /dir1

**Useradd student** => to create the user student

Su -student => to login to the user student.

Touch a1 a2 => permission denied because no write permission to user student.

**Chmod:**

To change the file or directory access permissions.

To give the permissions in symbolic value.

Chmod ugo+rwx <file or dir>

To give the permission in octal value.

Chmod 777 <filename or directory name>

Chmod o+w /dir1

su -student

cd /dir1

Touch a1 a2 a3

Ls

a1 a2 a3

**Do not allow the user student to each other into dir1**

Chmod o -x => remove the execute permission

Su -student

Cd /dir1 => permission denied

Chmod 777 /dir1 => the permissions applied to only dir1 and subdirectories and files.

Chmod -R 777 /dir1 => the permissions applied to all the directories and files under dir1.

**Chgrp:** To change the permissions to group.

Groupadd dba => create a group dba.

Mkdir /d1

Ls -ld /d1

For directory /d1 give the write permissions for all users under dba group.

Chgrp dba/d1

Ls -ld/d1

Chmod g+w /d1.

To create the users with dba group

Useradd -g dba u1

Useradd -g dba u2

Su -u1

Cd /dir1

Touch a s d f

To change the group dba for /d1 including all files and subdirectories

Chgrp -R dba /d1

Chown => to change the ownership permissions.

Give the write permission on file1 for user student.

Chown student file1

Ls -l file1

Su -student

Cat >> file1

This is my second line of fiel1

Ctrl + D

**Group Administration**

**Group:** A group is a collection of users who can share the files and the other system resources. A group consists of group name, and the list of users belong to that group.

These groups classified into 2 types:

1. Primary group

2. Secondary group

**Primary Group:** It’s a group that every user must belong to, without a primary group the user cannot created, and the user cannot have more than one primary group.

**Secondary Group:** It’s a group the user can also belongs, the user can belong to max 15 groups.

**Syntax:**

**group add -g <gid > <gname>**

0-499 => are system defined till version 6.0

500- 60000 ports are usable.

0 to 999 are system defined in ver 7.0

1000- 60000 ids are used.

**Groupadd -g 500 dba =>** Created the group with users defined group id.

**Groupadd -g dba** **=>** created the group with system defined id.

If -g <gid> is not defined system takes default id s from 1000.

**/etc/group =>** is the file contains all the group names and group ids.

**Cat /etc/group** => to see all group ids and names.

**Grep dba /dba/group** => to see the specified group called dba.

**Create group**

[#] groupadd devops

# cat /etc/group

# cat /etc/gshadow

// Devops group is created but no one is added.

How to add users to group?

# usermod -G devops vinodh

// I am adding vinodh to devops group

//now add andother user

# usermod -G devops ramu

// can add multiple users to the group devops

# useradd u1

# useradd u2

# useradd u3

# useradd u4

# usermod -G devops u4

// using usermod we can add only one user, but using gpasswd we can add one or multiple users.

# gpasswd –help

# gpasswd -a u1 devops

# gpasswd -M u2, u3 devops

**How to make user Admin?**

# gpasswd -A vinodh devops

# cat /etc/gshadow

// now again make u1 as admin to devops group

# gpasswd -A u1 devops

// now vinodh is override by u1 as admin.

//

# cat /etc/gshadow

[#] groupdel devops

[#] userdel vinodh

// If you delete the user vinodh the home directory and mailbox are not deleted.

[#]cd /home/vinodh

[vinodh #] cd

//it is not deleting the mailbox also

[#] cd /var/spool/mail

[mail #]ls

Vinodh u1 u2 u3

//if you want to delete the user permanently

[ #] userdel -r ramu

//the above command removing everything associated with the user ramu.

// Now create new user and update password

[#] Useradd rajesh

[#] passwd rajesh

Enter passwd: \*\*\*\*\*

[#] visudo

At last line

Rajesh ALL=(ALL) NOPASSWD: ALL

ALL=(ALL) => means all machines all commands.

**\*\*\*\*\*\*\*\*\*\*\*\***

Rajesh user already there but AWS is restricted him to login if you login directly.

(By default, it will look for public key and private key)

[$] sudo -i

[#] vi /etc/ssh/sshd\_config

In this file one password authentication is there uncomment and make it yes.

#PasswordAuthentication yes

# systemctl reload sshd\_config

// here we have to pass the service name not the config file name.

# systemctl reload sshd

// now you can login with user rajesh

$ ssh -i “devops.pem” [rajesh@ip.us-east-2.compute.amazonaws](mailto:rajesh@ip.us-east-2.compute.amazonaws)

Password: \*\*\*\*\*

[rajesh@\*\*\*]$ now rajesh user is connected directly.

Once login change the password immediately.

$ passwd

Changing password for user rajesh

Current password:

Newpassword: \*\*\*// strong password

// now password updated successfully.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Suppose user is resigned from company how to lock the user.

[#] usermod -L rajesh

[#] cat /etc/shadow

You can see here locked users also with !

**\*\*\*\*\*\*\*\*\***

[#] passwd -S rajesh

[#] usermod -u rajesh

[#] passwd -S rajesh

How many users logged in into the system?

[#] who

Who command tells us how many users are logged in.

// this way u can see how many users logged in to the particular system.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Now create another user,

[#] Useradd suman

[#] passwd suman

[#] visudo

Suman ALL=(ALL) ALL ALL=NOPASSWD

// no of days in which user has to change the password

[#] chage -d 0 suman

0 means password needs to be change immediately.

0 => password expiry.

[#] chage suman

// normally contract employees this will useful for 30 days and week like that.

[#] chage -l suman

// this will show the default values available to the user.

[#] chage suman

// you can change the parameters here.

[#] chage -l suman

[#] chage –help

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Ownership**

[$] touch file file1 file2 file3 file4

[$] ls -l

[$] sudo -i

[#] touch sample file1 file2 file3

[#] ls -l

[#] useradd vinodh

# groupadd devops

# chown vinodh sample

# ls -l

// you can see ownership changed

// Now you can change the group ownership of the file.

# chgrp devops sample

# touch file100

# ls -l

# chown vinodh: devops file100

# mkdir dir1

# ll

# chown vinodh:devops dir1

# chown -R vinodh:devops dir1

For a file or directory there is some default permissions

**Inode**

**Inode** -> is a data structure it will store file or directory information (size and permissions)

An inode doesn’t store the content of the file and filename.

Ls -li -> it displays with inode numbers

Ls -l -> listing along with size but not in human-readable format

Ls -lh -> listing along with size.

echo $HOME

pwd -> present working directory

cd ~ or cd ->Change to your home directory

cd .. -> change to present directory

cd - -> change to previous directory.

**Before get in to linux file system we will see the Linux users and their locations.**

Linux file system is case-sensitive. Hidden files start with a (.) extension

User and group administration related commands: -

Uid

Useradd

RHEL 7.x onwards UID starts from 1000

Before that version starts from 500

**Usermod L**

**Usermod l**

**Usermod g**

**Cp:** Moving the files from one dir to another dir

**What is the diff between cp and mv?**

File => checked based on content

Wc => counts the number of lines, words, bytes, or characters in a file.

Ln => It will create the link between files.

**What is the use of link files?**

Hard link soft link

If the permissions started with l It means the file is having a soft link.

**To find what OS using:**

Uname -s

Linux

Cat /etc/issue => to find what feature of OS we are using.

Echo $ SHELL => default shell

/bin/bash => to know the current shell

Sh => to change the shell.

Sh -4.1 # csh

Echo $0 => to know the current working shell

# exit

**reation of user UPG (user private group)**

User name: student /etc/passwd

Group name: student /etc/group

Home dir: student /home dir

Mail box: student /var/spool/mail dir

**To create password for user:**

**Passwd student**

**/etc/passwd** => file contains all users information

**/etc/shadow** => file contains all users password information

**The files related to irrespective of the user and the shell:**

**/etc/motd**

**/etc/profile**

**/etc/bashrc**

These files also called global profiles or global environment files.

**/etc/motd** => is the file form message of the day. The message in /etc/motd file execute on user console when user logged in. These messages only for new login but not for logged in users.

Vi /etc/motd

Good evening

Wq!

To see the welcome message:

Press ctrl+alt+F2

Ctrl+alt+F1 to see desktop version.

**/etc/profile** => The messages in /etc/profile is for all users. only new login users not logged in users.

Vi /etc/profile

Press esc+shift+g => to go to last line in page and last line

Echo welcome to India

To see messages ctrl+alt+F2

**/etc/bashrc** => the messages in /etc/bashrc file is only for new logins

Press esc shift g => to go to last page and last line.

Echo: have a nice day.

To change the umask value permanently or update /etc/bashrc file.

The default umask value for normal user is 002

The default umask value for root(super) user 022

**The file related with respective of users and the shell:**

.bash\_profile => execute when user logged in.

.bash\_logout => execute when user logged out.

.bash\_bashrc => for bash shell.

.bash\_history => maintains the history of all commands.

These files are also called user environment files or user profiles resides in every user home directory.

Etc/bashrc => global

.bashrc => local

The user default parameters are stored in /etc/default /useradd

# vi /etc/default/useradd

HOME = /home

INACTIVE = -1

EXPIRE =

SHELL = /BIN/BASH

SKEL = /ETC/SKEL

/etc/skel => is the directory contains the user environment files or user profiles the files under /etc/skel copied to the user home directory when the user is created with useradd command.

/etc/login.defs file contains the user default login values.

**Monitoring the Users**

Whoami => to know who is logged in.

Id => list the uid and gid of the current user

Id <username> => list the uid and gid of the current user of the specified user.

Last => when the server is last

Lastreboot => to see all reboots

Last<username> = > the user login and logout details.

Finger<username> => the user details information.

Who -i or -u => to see all logged in users with process ids?

Who or w or users => to see all logged in users

Lastlog => list the last logged in users

Cat /var/log/secure => to see the user login and logout details.

Cat /proc/cpuinfo => to see the cpu information.

History => to use the history of all commands.

History -c => to clear the history file

History -r => to restore the history file.

Last b => to see the user failed logins.

**Password aging:**

Chage student

**Group Modification:**

# groupmod [-n, -g]

# groupmod -g <new gid> <gname> => to change the group id

# groupmod -n <new gname> <old gname> => to change the group name

# groupdel <gname> => to delete the group

**User Modifications:**

#useradd [ -u, -g, -G, -d, -c, -m, -s, -l]

U=> to change userid

g => to change the primary group

G => to change the secondary group

c => to add or change the comment

d =>to change the user home directory

m => make directory

s => to change the shell

l => to change the username

#userdel <username> => to delete the user

# userdel -r <username> => to delete the user with home directory.

r => recursively.

**SUDO USERS:**

1. sudoers allows particular users to run various root user commands without needing a root password.

2. /etc/sudoers is the configuration file for sudoers to configure normal users as privileged users.

3. It is not recommended to open this file using vim editor because this editor cannot check the syntax.

An editor for sudo users is **visudo.**

4. Once the file is opened nobody can open this file on different terminal “the file is busy” message is displayed on the terminal for security reasons.

**How to give sudo permission to Normal user.**

#visudo => go to line 98

Student ALL = ALL

**Note:** If any syntax errors in the file, those error are displayed with line no’s, press ‘e’ to edit this file and save the file.

# su -student

Student $ sudo useradd user1

Student $ sudo useradd user2

Student$ exit

#

1. we can assign sudo permissions to no. of. users by simplying names separated by commas, or line by line instead of giving all permissions, we can give only same commands as follows,

Student ALL = /usr/sbin/useradd, /user/sbin/Usermod.

2. We can also apply for one group or groups as follows

%dba ALL = ALL or individual commands separated by commands.

3. We can also create one command alias add some commands to that alias and mention that alias to users.

Cmnd\_alias Networking = /usr/sbin/route, /usr/sbin/ifconfig

Student ALL = NETWORKING

4. We can also create one user alias and add the users to that alias and assign some commands to that alias.

User\_alias OURTEAM = student, Dave, Steve

OURTEAM ALL = ALL

5. In which location the sudo users commands history is logged .

6. All the sudo users commands history is logged in /var/log/secure file.

**Commands used so far:**

**Touch** –It will create a file with zero bytes

**Find** – find command is used to search and locate a list of files and directories based on conditions you specify for files that match the arguments. Find can be used in a variety of conditions like you can find files by permissions, users, groups, filetype, date, size, and other possible criteria.

**Umask** – **user mask or user file creation mask:** It is used to set the permissions for files/directories newly created on a Linux machine.

**Chmod:** It will change the file or directory access permissions

**Chown**: It will change the ownership of a file name. the only root user can execute this.

**Chgrp:** It will change the group ownership of a file or directory. The only root user can execute this.

**Cp:** It will copy the contents of a file to another file

**Mv:** It will move or rename a file

**File:** Determine file type

**Wc:** Counts the number of bytes, lines, words, or characters in a file.

**Ln:** It will create the link between files.

**Commands so far used:**

Useradd

passwd

visudo

usermod

groupadd

passwd

gpasswd

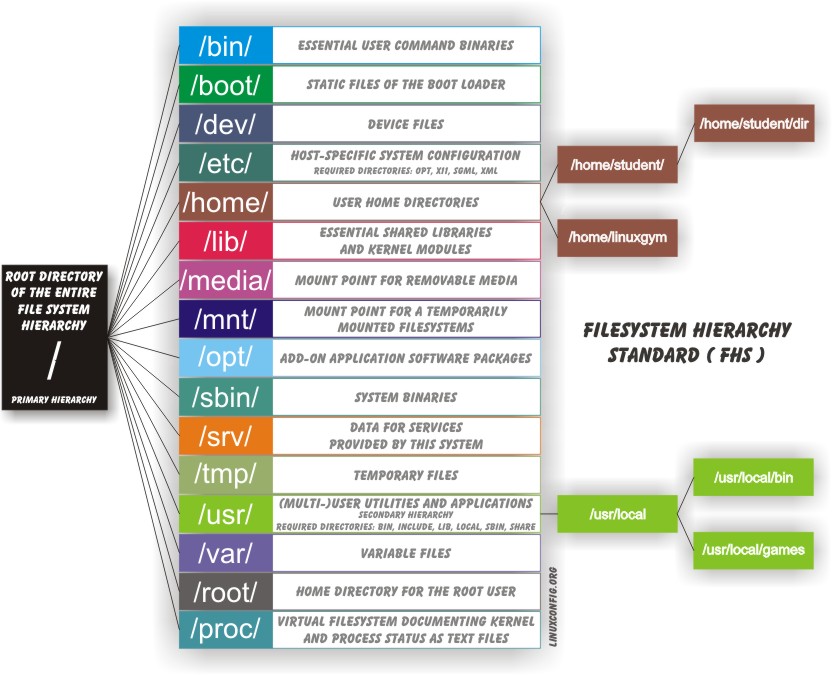
userdel

groupdel

**FHS (**F**ilesystem**H**ierarchy**S**tandard)**

Linux systems store their important files according to a standard layout called the **F**ilesystem **H**ierarchy **S**tandard (**FHS**), which has long been maintained by the Linux Foundation. For more information, take a look at the following document: *"*[*Filesystem Hierarchy Standard*](https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf)*"* created by LSB Workgroup. Having a standard is designed to ensure that users, administrators, and developers can move between distributions without having to re-learn how the system is organized.

Linux uses the ‘**/**’ character to separate paths (unlike Windows, which uses ‘**\**’), and does not have drive letters. Multiple drives and/or partitions are mounted as directories in the single filesystem. Removable media such as USB drives and CDs and DVDs will show up as mounted at **/run/media/yourusername/disklabel**for recent Linux systems, or under **/media** for older distributions. For example, if your username is **student** a USBpen drive labeled FEDORA might end up being found at **/run/media/student/FEDORA**, and a file **README.txt**on that disc would be at **/run/media/student/FEDORA/README.txt**.



**Linux Directory structure or Filesystem Hierarchy**

Filesystems can vary with how they are structured, but for the most part they should conform to the Filesystem Hierarchy Standard.

Go ahead and do a **ls -l /** to see the directories listed under the root directory, yours may look different than mine, but the directories should for the most part look like the following:

**Cd /**

**/ =>** The root directory of the entire filesystem hierarchy, everything is nestled under this directory.

**/ -> Root Directory**

**/root**

The root user's home directory.

**/boot**

The image of the kernel is in here,Contains kernel boot loader files.

**/etc**

Core system configuration directory, should hold only configuration files and not any binaries.

Important files here is:

**/etc/passwd** => This file tells you about the user that has been created, what is the format that the password is got stored, which shell and which is the home directory of the user.

**/etc/shadow** => Encrypted password of each and every user.

**/etc/group** => Information about each group of the system contains. (groupadd devops, now check in this file)

**/etc/grub/\*** => contains the information which kernal to be loaded

**/etc/inittab** => it starts of the system and initializes all other services.

**/home**

**Personal directories for users, holds your documents, files, settings, etc. All users stored in the home directory (in windows c drive users directory) Ex: Vinodh, Satvik, Milton**

**/mnt and /media**

both are same purpose they are a general-purpose mount point, If you want to attach a pendrive to the system, you want to see the contents of that pendrive, at that time you have to mount that pendrive into /mnt or /media. If it is GUI, it happens automatically.

**/proc**

**This file system does not have to be populated by the user. It stores all the kernal settings at runtime so every time the system is shutdown the proc file is deleted and when the system is restarted the proc file is recreated.**

Information about currently running processes. All the process id s available in **proc** directory. along with that it contains **cpuinfo** file which contains all the cpu information like how much **cpu utilized**, how much **ram size** used.

**/bin**

Essential ready-to-run programs (binaries), includes the most basic commands such as **ls** and **cp**.

**A**ll commands accessible by all users mkdir**, ls, date, etc.**

**/sbin**

System bin Contains essential system binaries, usually can only be run by root user, it will not accessible by normal users. Ex: Useradd, shutdown

**/var**

It contains variable files. Variable directory, it's used for system logging, user tracking, caches, etc.

Basically, anything that is subject to change all the time. This includes system log files (/var/logs), emails (var/mail), and temp **files needed across reboots**

**/var/log/\***

you can see lot of log files here.

**/opt**

By default, it is empty. we are going to install 3rd party software. (In windows program files. Optional application software packages: maven, sonarqube, jenkins, etc You have to install all software as a root user.

**/dev**

**Device files It** contains device files ex: USB or any device attached to the server. Special device files live here.Example printer device also available in dev directory.

**How many devices open xvda1 xvda2**

**Mounted on**

**/temp**

Temporary directory to store files,this directory can be accessed by any user. Normally we used this folder to switch files between root user directories and normal user directories.

**/lib**

Holds library files that binaries can use.

**/run**

Information about the running system since the last boot.

**/srv**

Site-specific data which are served by the system.

/**proc**

Process related commands, like

**/usr =>** unix sharable resourced, Third party softwares stored here.

**/usr/local/bin**=> binaries related to some third-party tools. jenkins terraform, python, maven all binaries stored here.

**/bin,**

This is unfortunately named, most often it does not contain user files in the sense of a home folder.

This is meant for user installed software and utilities, however that is not to say you can't add personal directories in there. Inside this directory are sub-directories for /usr/bin, /usr/local, etc.

**/sbin**

Used by administrators.

**lib (more executables and more libraries)**

**Net =>**

**Selinux =>**

**lost+found =>**

**sys =>**

**tmp =>**

**misc =>**

**sudoers, password, motd, cron. alow, cron. deny groups, grub/\*, inittab.**

**inode :**

Inodes store metadata such as:

* File type
* File size
* Owner ID
* Group ID
* Read, write and execute permissions
* Last access time
* Last change time
* Last modification time

**All inodes within the same filesystem are unique**. However, the same inode number can be used in different filesystems. Because the filesystem ID and each inode number are combined to create unique identification labels.

An inode is **a data structure that keeps track of all the files and directories within a Linux or UNIX-based filesystem**. So, every file and directory in a filesystem is allocated an inode, which is identified by an integer known as “inode number”. These unique identifiers store metadata about each file and directory.

*“Inode” is the abbreviation for “index node”.*

*Soft link and hard link l*