LET'S START LINUX

1. Linux Introduction

Linux is an open-source, Unix-like operating system (OS) kernel, initially developed by Linus Torvalds in 1991. It has since grown into a major operating system used for a wide range of computing tasks, from personal computers to servers, embedded devices, and supercomputers.

Key Features of Linux:

1. **Open Source:** The source code is freely available for anyone to view, modify, and distribute under the GNU General Public License (GPL).
2. **Multi-User System:** Multiple users can access the system simultaneously without interfering with each other's processes.
3. **Multitasking:** Linux efficiently handles multiple tasks at once, making it suitable for server environments.
4. **Security:** Strong security features include user privileges, access control, and file permissions.
5. **Portability:** Linux can run on a variety of hardware, from desktops and laptops to embedded systems and servers.
6. **Shell and Command Line Interface (CLI):** Linux offers powerful command-line utilities, enhancing control and automation.

Linux principles

- Everything is a file (including hardware)
- Small, single-purpose programs
- Ability to chain programs together to perform complex tasks
- Avoid captive user interfaces
- Configuration data stored in text

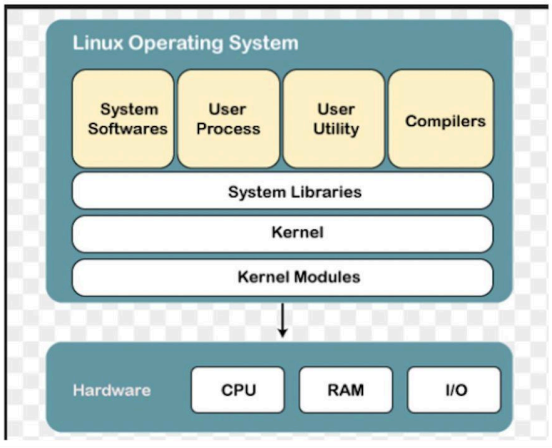
Why Linux?

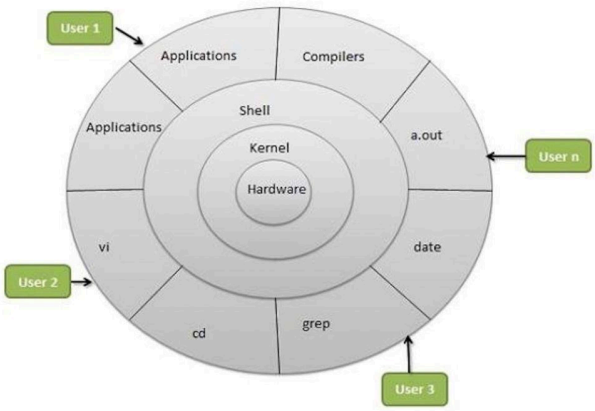
- OpenSource.
- Community support.
- Heavily customizable.
- Most Servers run on Linux.
- DevOps most of the tools are implemented on Linux only.
- Automation
- Secure

Components of a Linux System:

1. **Kernel:** The core part of Linux that interacts directly with hardware and manages system resources.
2. **Shell:** The command-line interpreter that allows users to execute commands and scripts.
3. **File System:** Hierarchical structure where files are organized, starting from the root directory (/).
4. **Graphical User Interface (GUI):** While Linux is known for its command-line capabilities, many distributions come with GUIs like GNOME, KDE, or XFCE.
5. **Package Manager:** Helps users install, update, and manage software. Examples include apt (for Debian/Ubuntu), dnf/yum (Fedora/CentOS), and pacman (Arch).

Linux Architecture

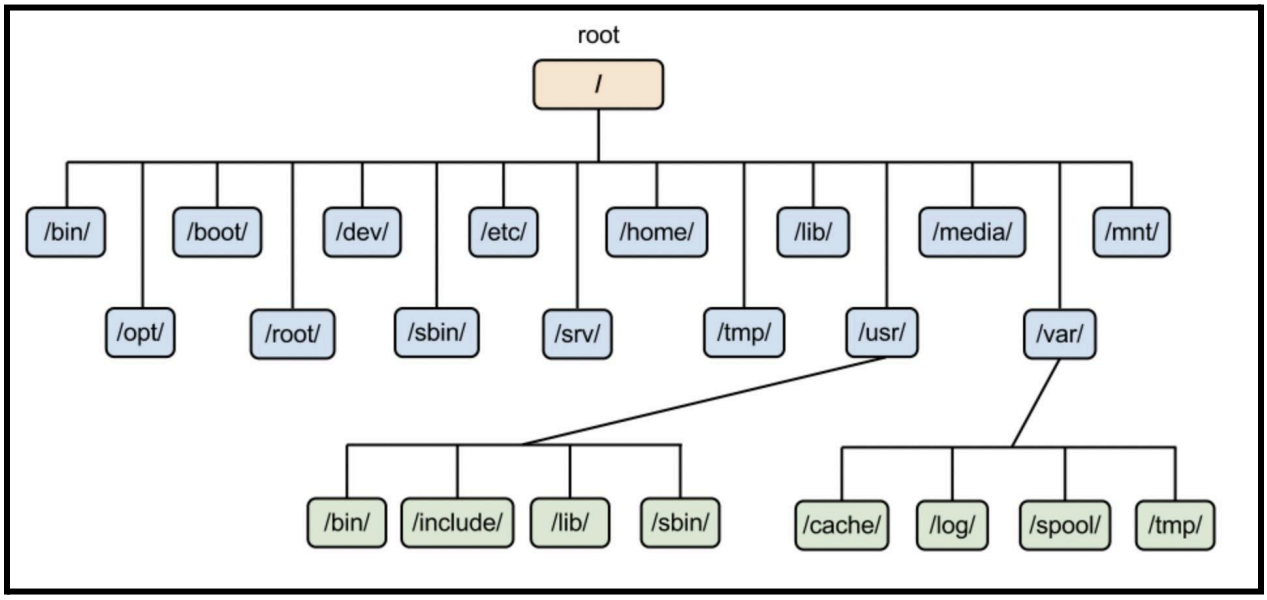




Some Important Directories

- **Home Directories:** /root,/home/username
- **User Executable:** /bin, /usr/bin, /usr/local/bin
- **System Executables:** /sbin, /usr/sbin, /usr/local/sbin
- **Other Mountpoints:** /media, /mnt
- **Configuration:** /etc
- **Temporary Files:** /tmp
- **Kernels and Bootloader:** /boot
- **Server Data:** /var, /srv
- **System Information:** /proc, /sys
- **Shared Libraries:** /lib, /usr/lib, /usr/local/lib

Linux Directory Structure



Different Linux distros

→ **Popular Desktop Linux OS**

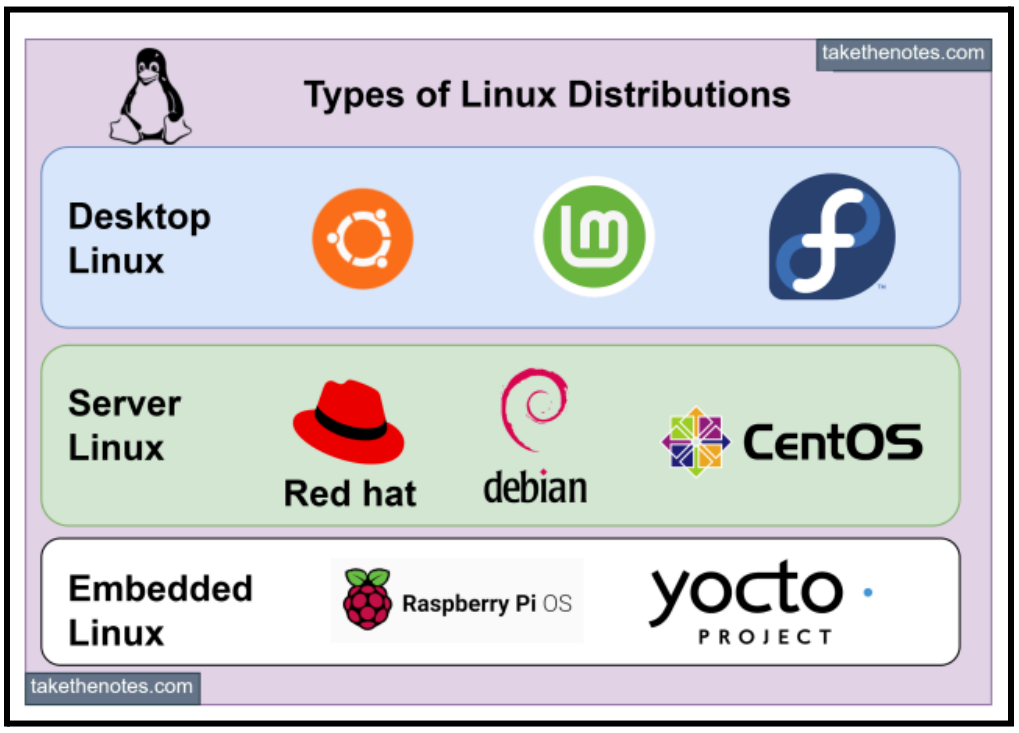
- Ubuntu Linux
- Linux Mint
- Arch Linux
- Fedora
- Debian
- OpenSuse

→ **Popular Server Linux OS**

- Red Hat Enterprise Linux
- Ubuntu Server
- Centos
- SUSE Enterprise Linux

Most used Linux distros currently in the IT industry

RPM based:- RHEL & Centos
Debian based :- Ubuntu Server



Difference between RPM based and Debian based.

From the user's point of view, there isn't much difference in these tools. The RPM and DEB formats are both just archive files, with some metadata attached to them. They are both equally arcane, have hardcoded install paths and only differ in subtle details. DEB files are installation files for Debian based distributions. RPM files are installation files for Red Hat based distributions. Ubuntu is based on Debian's package manage based on APT and DPKG. Red Hat, CentOS and Fedora are based on the old Red Hat Linux package management system, RPM.

DEB or .deb (Debian based softwares)
DEB is the extension of the Debian software package format and the most often used name for such binary packages. DEB was developed by Bedian.

Example: Google chrome software
Package name: google-chrome-stable_current_amd64**.deb**
Installation: dpkg -i google-chrome-stable_current_amd64**.deb**

RPM or .rpm (Red Hat based softwares.)
It is a package management system. The name RPM variously refers to the .rpm file format, files in this format, software packaged in such files, and the package manager itself. RPM was intended primarily for Linux distributions; the file format is the baseline package format of the Linux Standard Base. RPM was developed by Community & **Red Hat**.

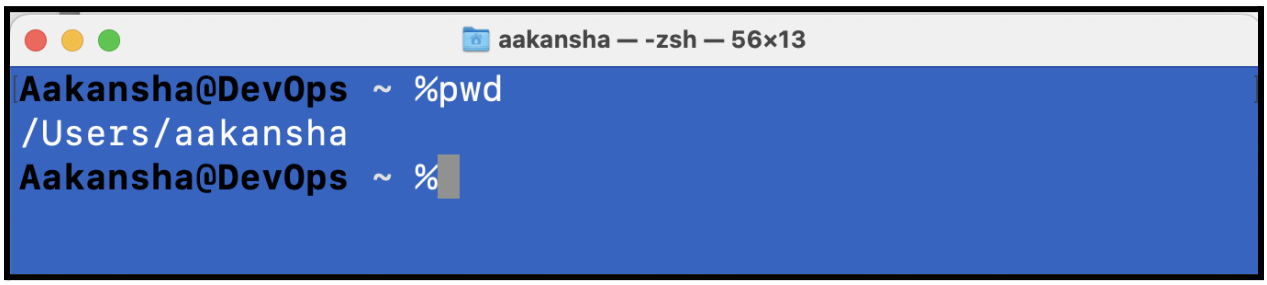
Example: Google chrome software
Package Name: google-chrome-stable-57.0.2987.133-1.x86_64.rpm
Installation: rpm -ivh google-chrome-stable-57.0.2987.133-1.x86_64.rpm

NOTE: You will also encounter different commands, packages and service names while using both kinds of distros.

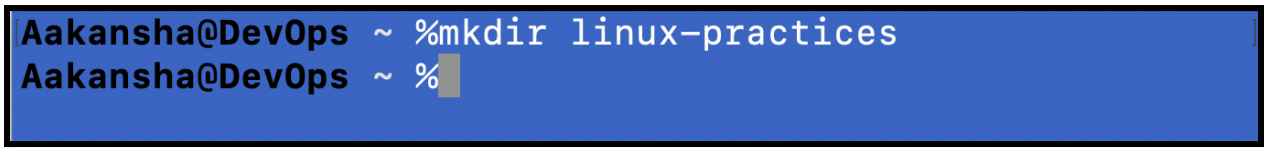
2. Basic Commands

➔ Open Terminal

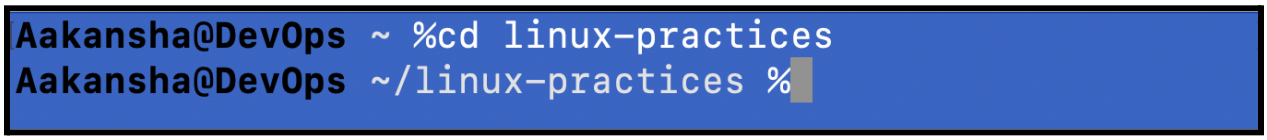
➔ Know where you are? Present Working Directory



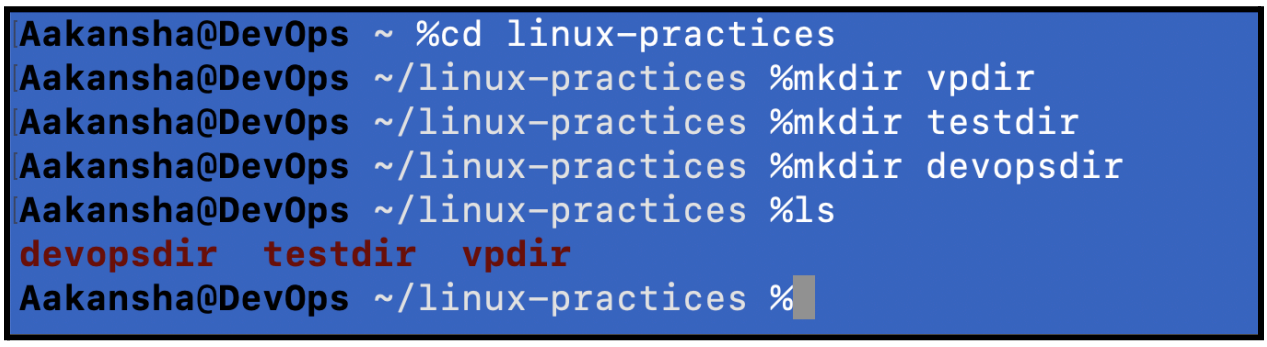
➔ Create a directory/folder in your home directory.



➔ Change your current working directory to linux-practices(Go to linux-practices folder).



➔ Create some more directories and list them with “ls” command.



➔ **Create some empty files with “touch” command and list them.**

```
Aakansha@DevOps ~/linux-practices %touch file1 file2 file3 file4
Aakansha@DevOps ~/linux-practices %ls
devopsdir  file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %
```

➔ **Reconfirm your location in your system.**

```
Aakansha@DevOps ~/linux-practices %pwd
/Users/aakansha/linux-practices
Aakansha@DevOps ~/linux-practices %
```

Absolute path and Relative path What is a path?

A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters.

What is an absolute path?

An absolute path is defined as the specification of the location of a file or directory from the root directory(/). In other words we can say absolute path is a complete path from the start of the actual filesystem from / directory.

Some examples of absolute path:

/Users/aakansha/linux-practices
/var/ftp/pub
/etc/samba.smb.conf
/boot/grub/grub.conf

If you see all these paths started from / directory which is a root directory for every Linux/Unix machine.

What is the relative path?

Relative path is defined as path related to the present working directory(pwd). Suppose I am located in /Users/aakansha and I want to change directory to /Users/aakansha/linux-practices. I can use the relative path concept to change directory to linux-practices and also devopsdir directory.

```
Aakansha@DevOps ~ %pwd
/Users/aakansha
Aakansha@DevOps ~ %cd linux-practices
Aakansha@DevOps ~/linux-practices %ls
devopsdir  file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %pwd
/Users/aakansha/linux-practices
Aakansha@DevOps ~/linux-practices %cd devopsdir
Aakansha@DevOps ~/linux-practices/devopsdir %pwd
/Users/aakansha/linux-practices/devopsdir
Aakansha@DevOps ~/linux-practices/devopsdir %
```

If you see all these paths did not start with / directory.

➔ Creating directories in devopsdir directory with absolute and relative path.

```
Aakansha@DevOps ~/linux-practices %ls
devopsdir  file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %mkdir devopsdir/ansible
Aakansha@DevOps ~/linux-practices %mkdir /Users/aakansha/linux-practices/devopsdir/aws
Aakansha@DevOps ~/linux-practices %ls devopsdir
ansible  aws
Aakansha@DevOps ~/linux-practices %
```

➔ Copying files into directory.

```
Aakansha@DevOps ~/linux-practices %pwd
/Users/aakansha/linux-practices
Aakansha@DevOps ~/linux-practices %ls
devopsdir  file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %cp file1 testdir
Aakansha@DevOps ~/linux-practices %cd testdir
Aakansha@DevOps ~/linux-practices/testdir %ls
file1
Aakansha@DevOps ~/linux-practices/testdir %
```

➔ **Copying directories from one location to another.**

```
Aakansha@DevOps ~/linux-practices %ls
devopsdir  file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %cp -R testdir/. vpdir/

Aakansha@DevOps ~/linux-practices %ls vpdir
file1  newfile1  newfile2
Aakansha@DevOps ~/linux-practices %ls testdir
file1  newfile1  newfile2
Aakansha@DevOps ~/linux-practices %
```

➔ **Moving files from one location to another.**

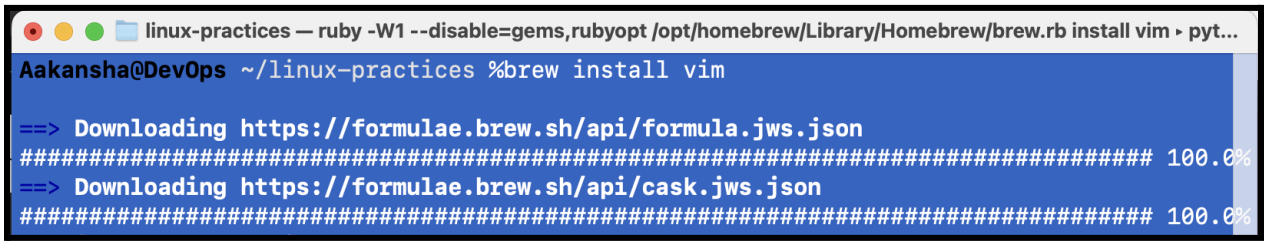
```
Aakansha@DevOps ~/linux-practices %pwd
/Users/aakansha/linux-practices
Aakansha@DevOps ~/linux-practices %ls
devopsdir  file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %mv devopsdir/ vpdir/
Aakansha@DevOps ~/linux-practices %ls
file1  file2  file3  file4  testdir  vpdir
Aakansha@DevOps ~/linux-practices %ls vpdir
devopsdir  file1  newfile1  newfile2
Aakansha@DevOps ~/linux-practices %mv file3 file4 vpdir
Aakansha@DevOps ~/linux-practices %ls
file1  file2  testdir  vpdir
Aakansha@DevOps ~/linux-practices %ls vpdir
devopsdir  file1  file3  file4  newfile1  newfile2
Aakansha@DevOps ~/linux-practices %
```

➔ **Removing files and directories.**

```
Aakansha@DevOps ~/linux-practices %ls
file1  file2  testdir  vpdir
Aakansha@DevOps ~/linux-practices %rm file1
Aakansha@DevOps ~/linux-practices %ls
file2  testdir  vpdir
Aakansha@DevOps ~/linux-practices %rm -rf testdir
Aakansha@DevOps ~/linux-practices %ls
file2  vpdir
```

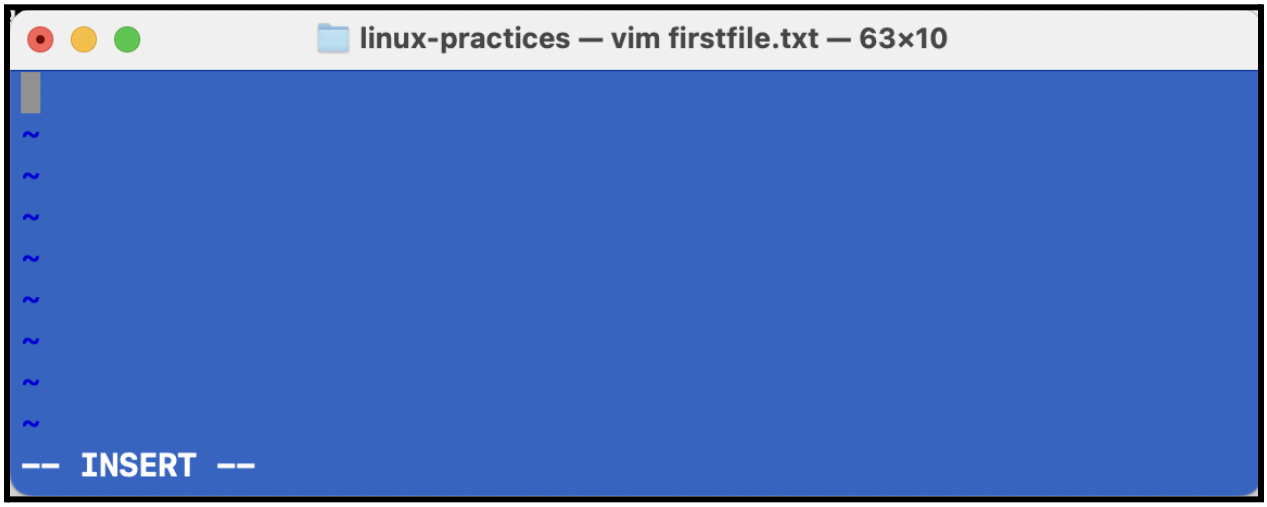
3. VIM EDITOR

➔ **Install vim editor.**

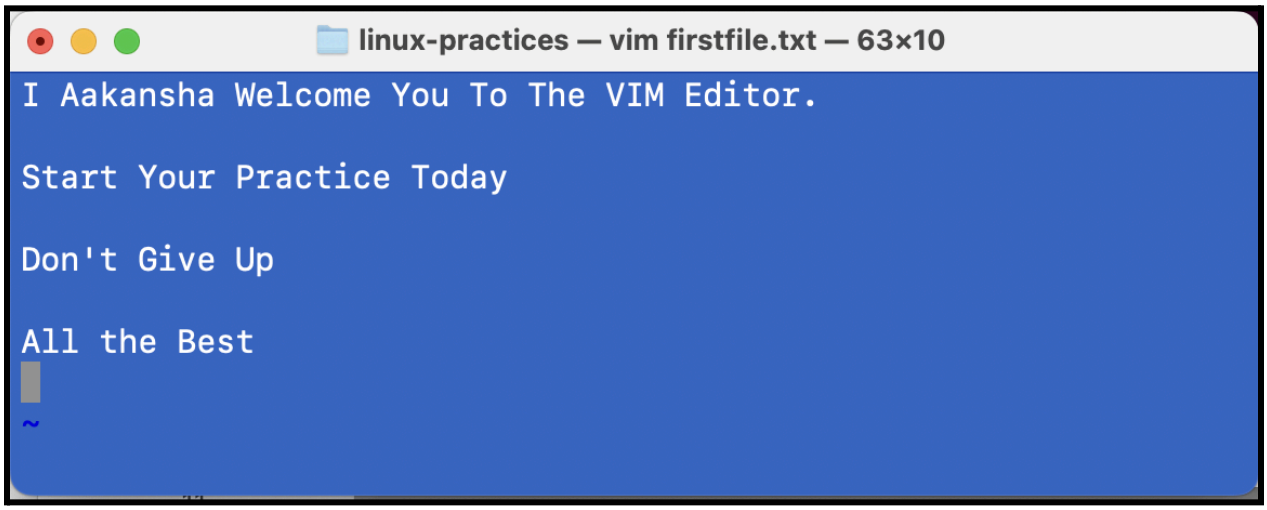


➔ **Open up a file in vim editor**

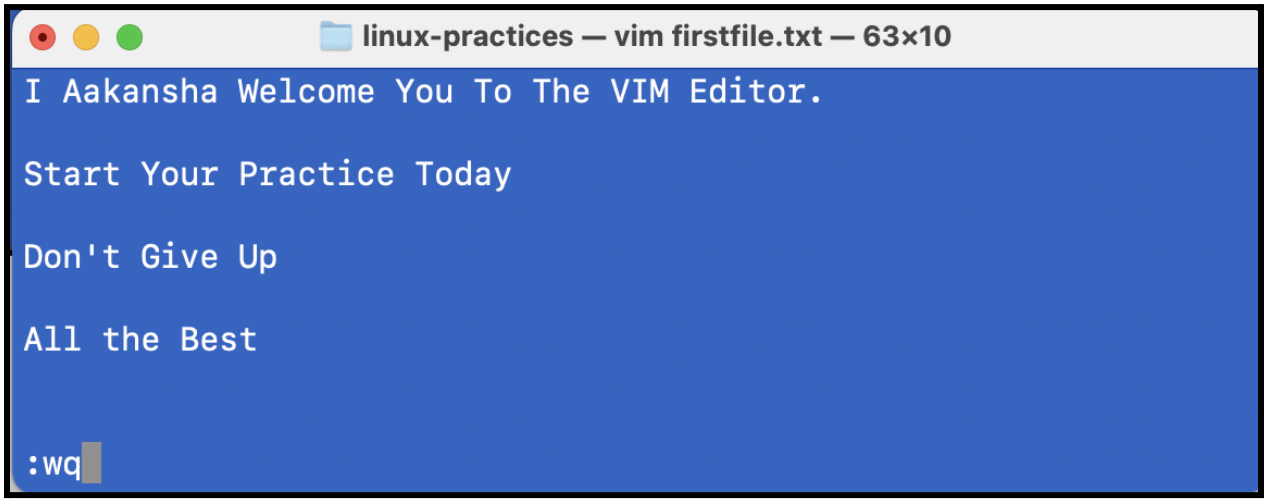
➔ **Hit i to enter into insert mode**



=> type few lines => hit Esc



=> type :wq



=> Enter.

➔ Read file with cat command.

```
Aakansha@DevOps ~/linux-practices %ls
firstfile.txt
Aakansha@DevOps ~/linux-practices %cat firstfile.txt
I Aakansha Welcome You To The VIM Editor.

Start Your Practice Today

Don't Give Up

All the Best
```

VIM EDITOR

VI Visual display editor
VIM Visual display editor improved

This is a command mode editor for files. Other editors in Linux are emacs, gedit vi editor is most popular.

It has 3 modes:

1. Command Mode
2. Insert mode (edit mode)
3. Extended command mode

Note: When you open the vim editor, it will be in the command mode by default.

Command Mode:

gg	To go to the beginning of the page
G	To go to end of the page
w	To move the cursor forward, word by word
b	To move the cursor backward, word by word
nw	To move the cursor forward to n words **(SW)**
nb	To move the cursor backward to n words **{SB)**
u	To undo last change (word)

Insert mode:

u	To undo the previous changes (entire line)
Ctrl+R	To redo the changes
VY	To copy a line
nyy	To copy n lines (Syy or 4yy)
p	To paste line below the cursor position
p	To paste line above the cursor position
dw	To delete the word letter by letter {like Backspace}
x	To delete the world letter by letter (like DEL Key)
dd	To delete entire line
ndd	To delete n no. of lines from cursor position{Sdd)
I	To search a word in the file

Extended command mode

Extended Mode is used for save and quit or save without quit using "Esc" Key with":"

Esc+:w	To Save the changes
Esc+:q	To quit (Without saving)
Esc+:wq	To save and quit
Esc+:w!	To save forcefully
Esc+wq!	To save and quit forcefully
Esc+:x	To save and quit
Esc+:X	To give passw or d to the file and remove password
Esc+:20(n)	To go to line no 20 or n
Esc+: se nu	To set the line numbers to the file
Esc+:se nonu	To Remove the set line numbers

ls command options

Options	Description
-l	Long listing format of files and directories, one per line
-a	List all hidden files and directories started with '.'
-F	Add a '/' classification at the end of each directory
-g	List all files and directories with the group name
-i	Print index number of each file and directory
-m	List all files and directories separated by comma ','
-n	List numeric UID and GID of owner and groups
-r	List all files and directories in reverse order
-R	Short list all directories
-t	Sorted by modified time, starting with the newest file

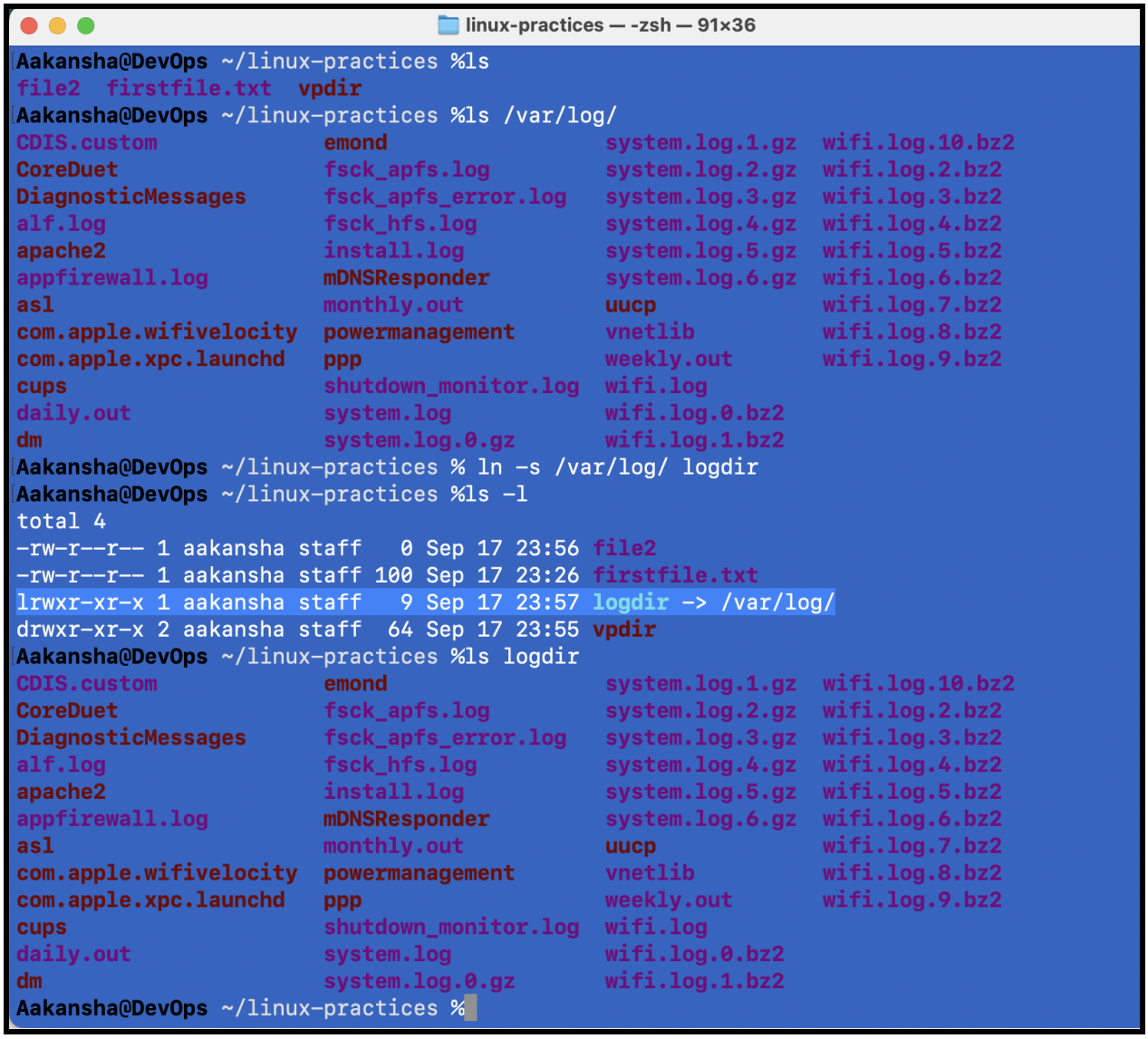
Types of files in linux

File Type	File Listing	Description
Regular file	`–`	Normal files such as text, data, or executable files
Directory	`d`	Files that are lists of other files
Link	`l`	A shortcut that points to the location of the actual file
Special file	`c`	Mechanism used for input and output, such as files in `/dev`
Socket	`s`	A special file that provides inter-process networking protected by the file system's access control
Pipe	`p`	A special file that allows processes to communicate with each other without using network socket semantics

Symbolic links

Symbolic links are like desktop shortcuts we use in windows.

Create a soft link for /var/log directory in our current working directory.

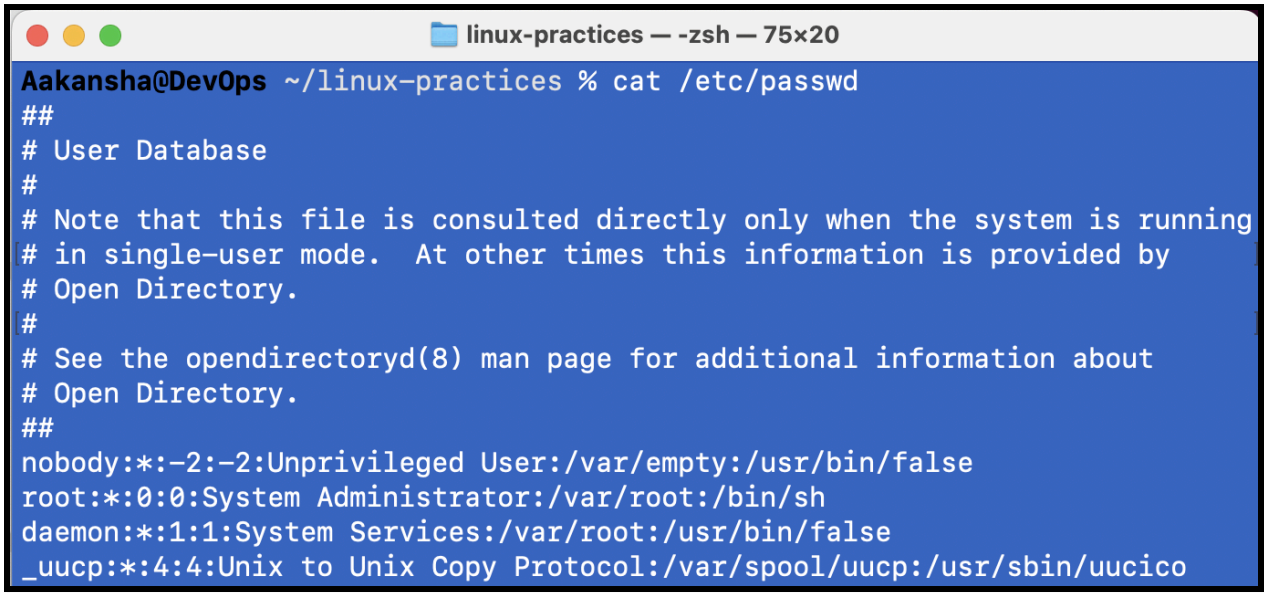


4. Filter & IO redirection command

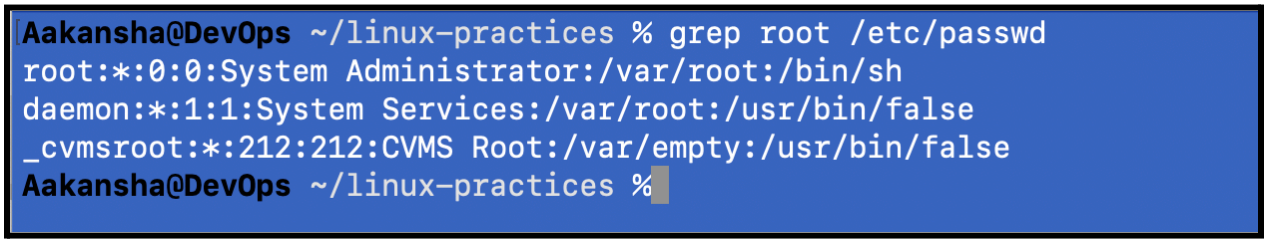
Grep

grep command is used to find texts from any text input.

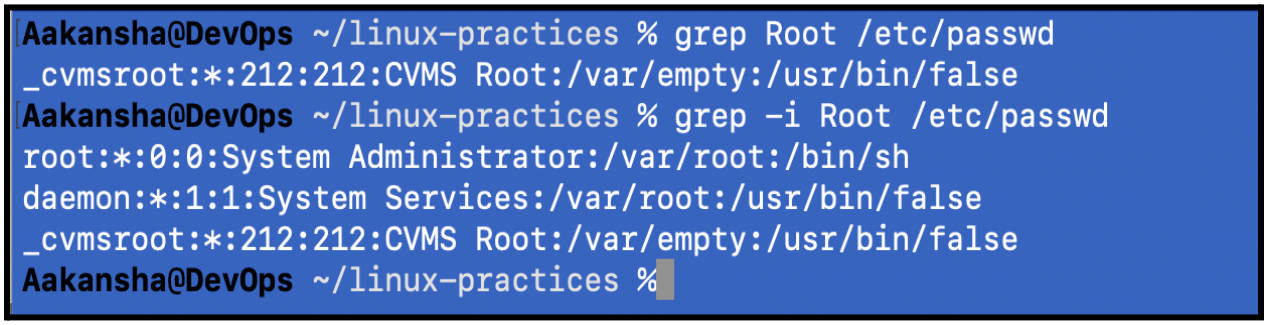
Passwd file: stores information about all the users in the system



➔ Finding line which contains word as “root” from /etc/passwd file.



➔ Linux is case sensitive, Root is different then root. Ignoring case in grep with -i option.



➔ To display things except the given word use -v option

Filter Commands

- **less:** Displays file content page wise or line wise.
- **Ex: less /etc/passwd**

Note: - press Enter key to scroll down line by line (or)

- **Used to go to next page**
- **Use b to go to previous page**
- **Use / to search for a word in the file**
- **Use v to go vi mode where you can edit the file in command**
- **once you save it you will back to less**

- **More: more is exactly same like less**
- **Ex: #more /etc/passwd**

Note: -press Enter key to scroll down line by line (or)

- **Used to go to next page**
- **Use / to search for a word in the file**
- **Use v to go vi mode where you can edit the file and once you save it you will back to more command**

- **Head: It is used to display the top 10 lines of the file.**
- **Ex:# head /etc/passwd**

```
Aakansha@DevOps ~/linux-practices % head /etc/passwd
##
# User Database
#
# Note that this file is consulted directly only when the system is running
# in single-user mode.  At other times this information is provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information about
# Open Directory.
##
Aakansha@DevOps ~/linux-practices %
```

- **Tail: It is used to display the last 10 lines of the file.**
- **Ex:# tail /etc/passwd**

```
Aakansha@DevOps ~/linux-practices % tail /etc/passwd
_demod:*:275:275:Demo Daemon:/var/empty:/usr/bin/false
_rmd:*:277:277:Remote Management Daemon:/var/db/rmd:/usr/bin/false
_accessoryupdater:*:278:278:Accessory Update Daemon:/var/db/accessoryupdate
r:/usr/bin/false
_knowledgegraphd:*:279:279:Knowledge Graph Daemon:/var/db/knowledgegraphd:/
usr/bin/false
_coreml:*:280:280:CoreML Services:/var/empty:/usr/bin/false
_sntpd:*:281:281:SNTP Server Daemon:/var/empty:/usr/bin/false
_trustd:*:282:282:trustd:/var/empty:/usr/bin/false
_darwindaemon:*:284:284:Darwin Daemon:/var/db/darwindaemon:/usr/bin/false
_notification_proxy:*:285:285:Notification Proxy:/var/empty:/usr/bin/false
_oahd:*:441:441:OAH Daemon:/var/empty:/usr/bin/false
Aakansha@DevOps ~/linux-practices %
```

- **cut**
- **# cut -d -f filename (where d stands for delimiter ex. :," "etc and f stands for field)**

```
Aakansha@DevOps ~/linux-practices % cut -d: -f1 /etc/passwd
##
# User Database
#
# Note that this file is consulted directly only when the system is running
# in single-user mode.  At other times this information is provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information about
# Open Directory.
##
nobody
root
daemon
```

To delimit spaces and print the field

#cut -d " " -f1 filename

I/O redirection

I/O redirection is a technique in Unix-like operating systems, such as Linux and macOS, used to redirect the input or output of commands from standard locations (keyboard and screen) to other files or devices. It allows users to control where data comes from and where it goes.

Here's an overview of common I/O redirection symbols:

Symbol	Description
`>`	Redirects the standard output (stdout) of a command to a file, overwriting the file if it exists.
`>>`	Redirects the standard output (stdout) of a command to a file, appending to the file if it exists.
`<`	Redirects the standard input (stdin) to take input from a file instead of the keyboard.
`2>`	Redirects the standard error (stderr) output to a file, overwriting the file if it exists.
`2>>`	Redirects the standard error (stderr) output to a file, appending to the file if it exists.
`&>`	Redirects both stdout and stderr to the same file, overwriting the file if it exists.
`&>>`	Redirects both stdout and stderr to the same file, appending to the file if it exists.

Examples:

1. **Redirecting stdout to a file:**

```
ls > output.txt
```

This command will redirect the output of `ls` to the file `output.txt`.

2. **Appending stdout to a file:**

```
echo "Hello" >> output.txt
```

This will append the string "Hello" to `output.txt`.

3. **Redirecting stderr to a file:**

```
ls non_existing_file 2> error.log
```

This will redirect the error message from `ls` to the file `error.log`.

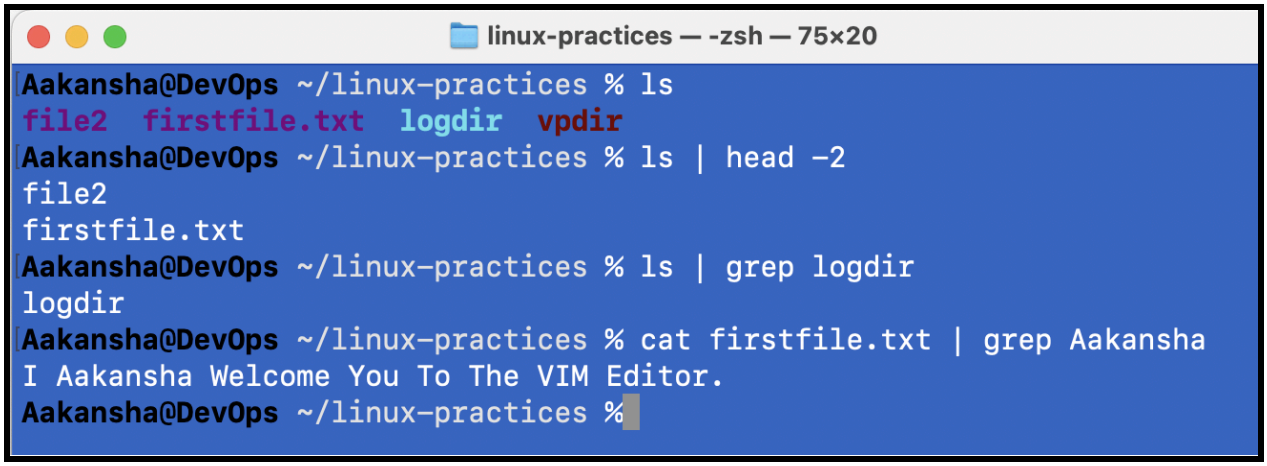
4. **Redirecting stdout and stderr to the same file:**

```
command &> output_and_error.txt
```

This will redirect both the output and any errors of `command` to `output_and_error.txt`.

Piping

So far we've dealt with sending data to and from files. Now we'll take a look at a mechanism for sending data from one program to another. It's called piping and the operator we use is (|). What this operator does is feed the output from the program on the left as input to the program on the right.



Find

The `find` command in Linux and Unix-based systems is used to search for files and directories in a directory hierarchy. It allows users to search for files based on various criteria like name, size, modification time, permissions, etc. It's a powerful tool for locating files and performing actions on them.

Basic Syntax:

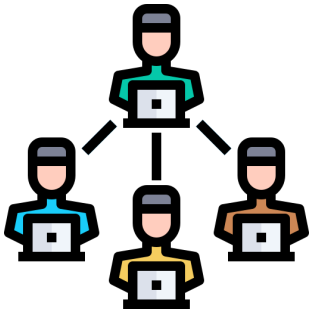
```
find [path] [options] [expression]
```

- **path:** The directory where you want to start searching (e.g., `/home/user`).
- **options:** Various flags to customize the search (e.g., `-name`, `-type`).
- **expression:** Conditions or actions to perform (e.g., searching by file name, size, etc.).

Commonly Used Options and Examples:

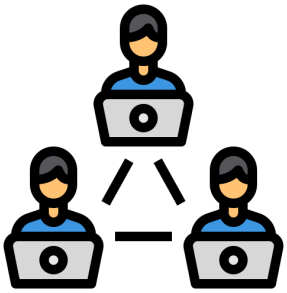
Option	Description	Example
`-name`	Search for a file with a specific name (case-sensitive).	`find /home -name "file.txt"` – Search for `file.txt` in `/home` directory.
`-iname`	Search for a file with a specific name (case-insensitive).	`find /home -iname "file.txt"` – Case-insensitive search for `file.txt`.
`-type`	Search for a specific type of file (`f` for file, `d` for directory).	`find /home -type d` – Search for directories in `/home`.
`-size`	Search for files based on their size.	`find /home -size +10M` – Find files larger than 10 MB.
`-user`	Search for files owned by a specific user.	`find /home -user username` – Search for files owned by `username`.
`-mtime`	Search for files modified a specific number of days ago.	`find /home -mtime -7` – Find files modified within the last 7 days.
`-perm`	Search for files with specific permissions.	`find /home -perm 644` – Find files with `644` permissions.
`-exec`	Execute a command on each file found.	`find /home -name "*.log" -exec rm {} \;` – Find all `.log` files and delete them.
`-delete`	Delete files directly that match the search criteria.	`find /home -name "*.tmp" -delete` – Find and delete all `.tmp` files.
`-empty`	Search for empty files or directories.	`find /home -empty` – Find all empty files or directories in `/home`.

5. Users & Groups



GROUP- 1

USERS



GROUP- 2

Some Important Points related to Users:

- Users and groups are used to control access to files and resources
- Users login to the system by supplying their username and password
- Every file on the system is owned by a user and associated with a group
- Every process has an owner and group affiliation, and can only access the resources its owner or group can access.
- Every user of the system is assigned a unique user ID number (the UID)
- Users name and UID are stored in /etc/passwd
- User's password is stored in /etc/shadow in encrypted form.
- Users are assigned a home directory and a program that is run when they login (Usually a shell)
- Users cannot read, write or execute each other's files without permission.

Types of user

TYPE	EXAMPLE	USER ID (ID)	GROUP ID (GID)	HOME DIR	SHELL
ROOT	**root**	**0**	**0**	**/root**	**/bin/bash**
REGULAR	**aakansha, vagrant**	**1000 to 60000**	**1000 to 60000**	**/Users/aakansha**	**/bin/bash**
SERVICE	**ftp, ssh, apache**	**1 to 999**	**1 to 999**	**/var/ftp etc**	**/sbin/nologin**

In Linux there are three types of users

1. Super user or root user
Super user or the root user is the most powerful user. He is the administrator user.

2. System user
System users are the users created by the softwares or applications. For example if we install Apache it will create a user apache. These kinds of users are known as system users.

3. Normal user
Normal users are the users created by root user. They are normal users like Rahul, Musab etc. Only the root user has the permission to create or remove a user.

Whenever a user is created in Linux things created by default:-

- A home directory is created(/home/username)
- A mailbox is created(/var/spool/mail)
- Unique UID & GID are given to user

Passwd file

1. /etc/passwd

The above fields are

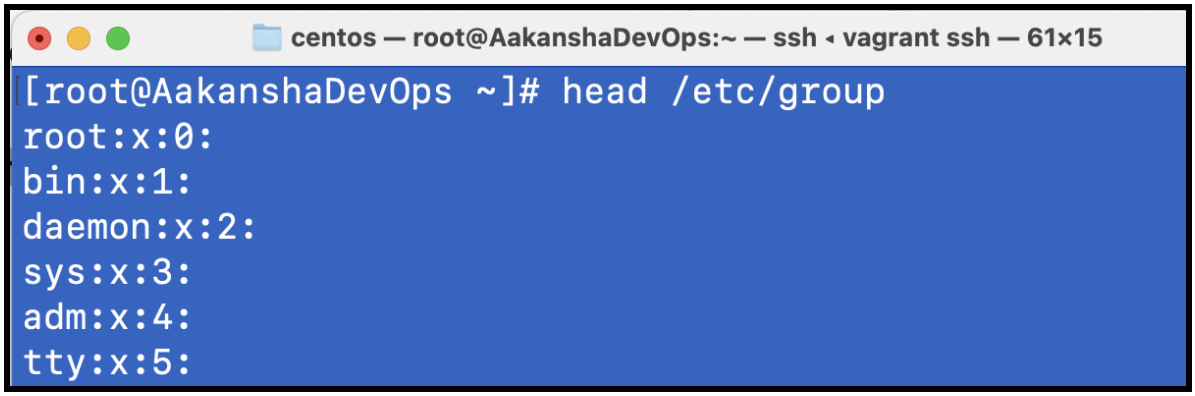
- **root** = name
- **x** = link to password file i.e. /etc/shadow
- **0** or **1** = UID (user id)
- **0** or **1** = GID (group id)
- **root** or **bin** = comment (brief information about the user)
- **/root** or **/bin** = home directory of the user
- **/bin/bash** or **/sbin/nologin** = shell

Group file

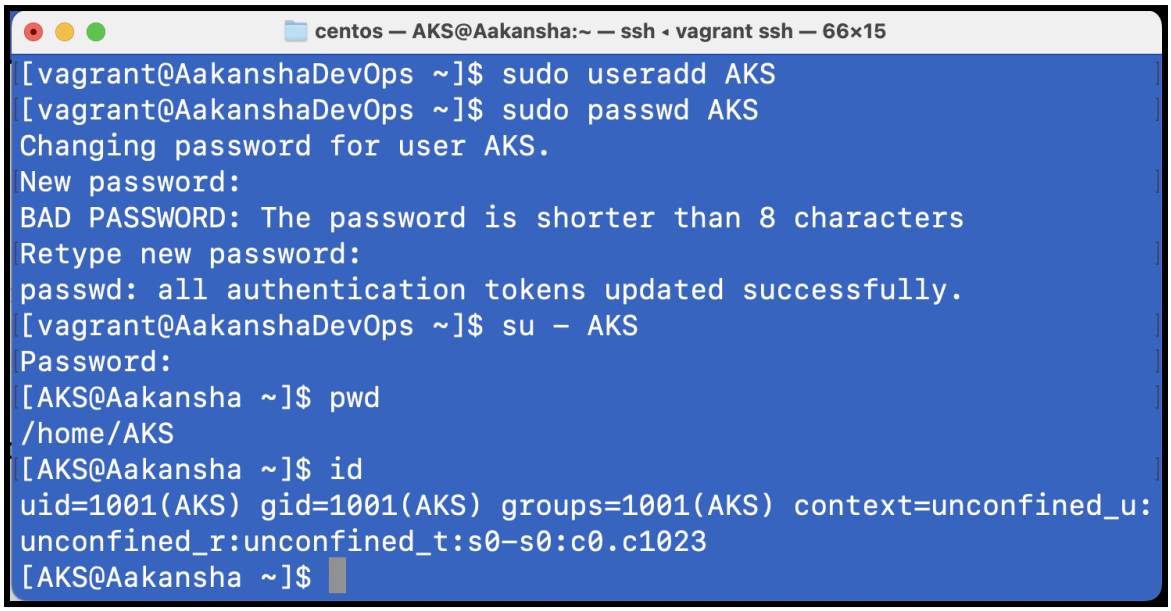
2. /etc/group

The file /etc/group stores group information. Each line in this file stores one group entry.

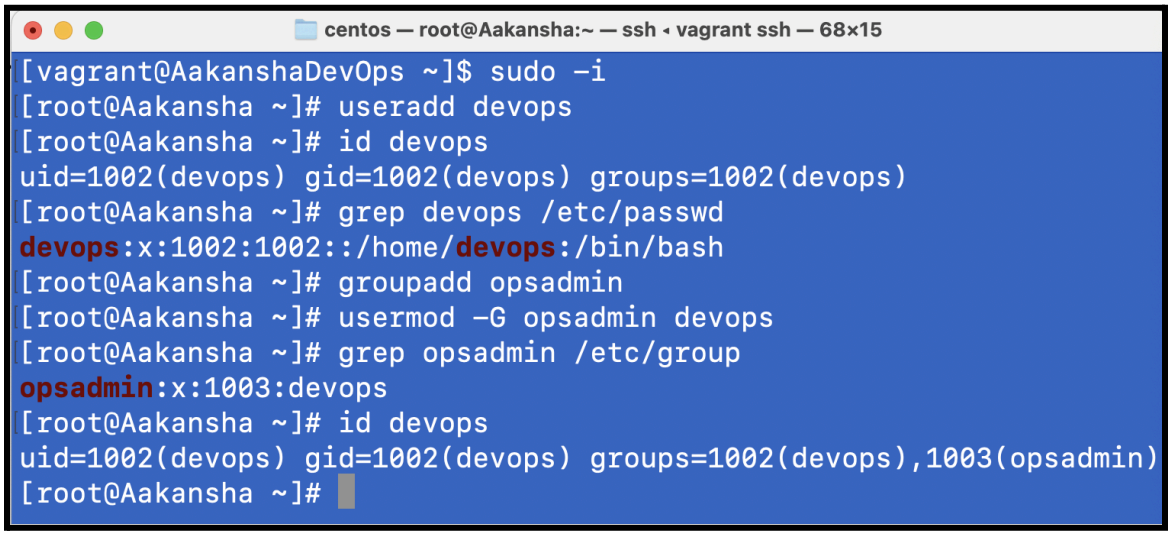
Group name, group password, GID, group members



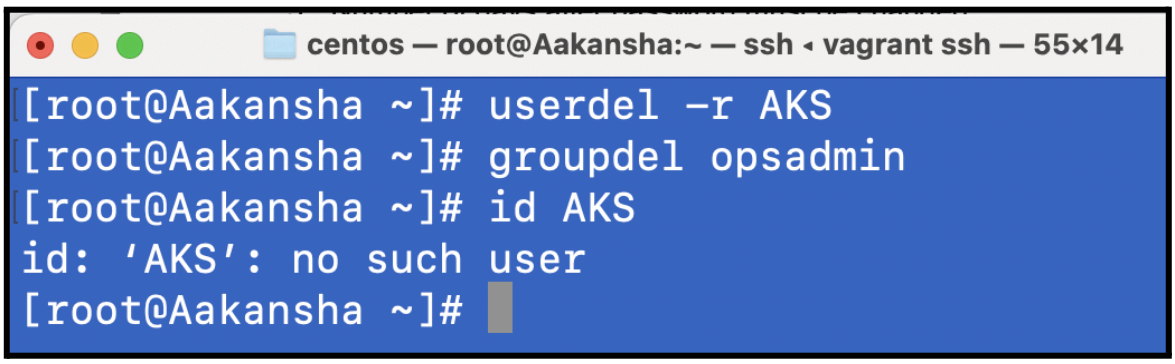
ADD USER, SET PASSWORD & SWITCH TO USER



ADD USER, GROUP & USER INTO GROUP



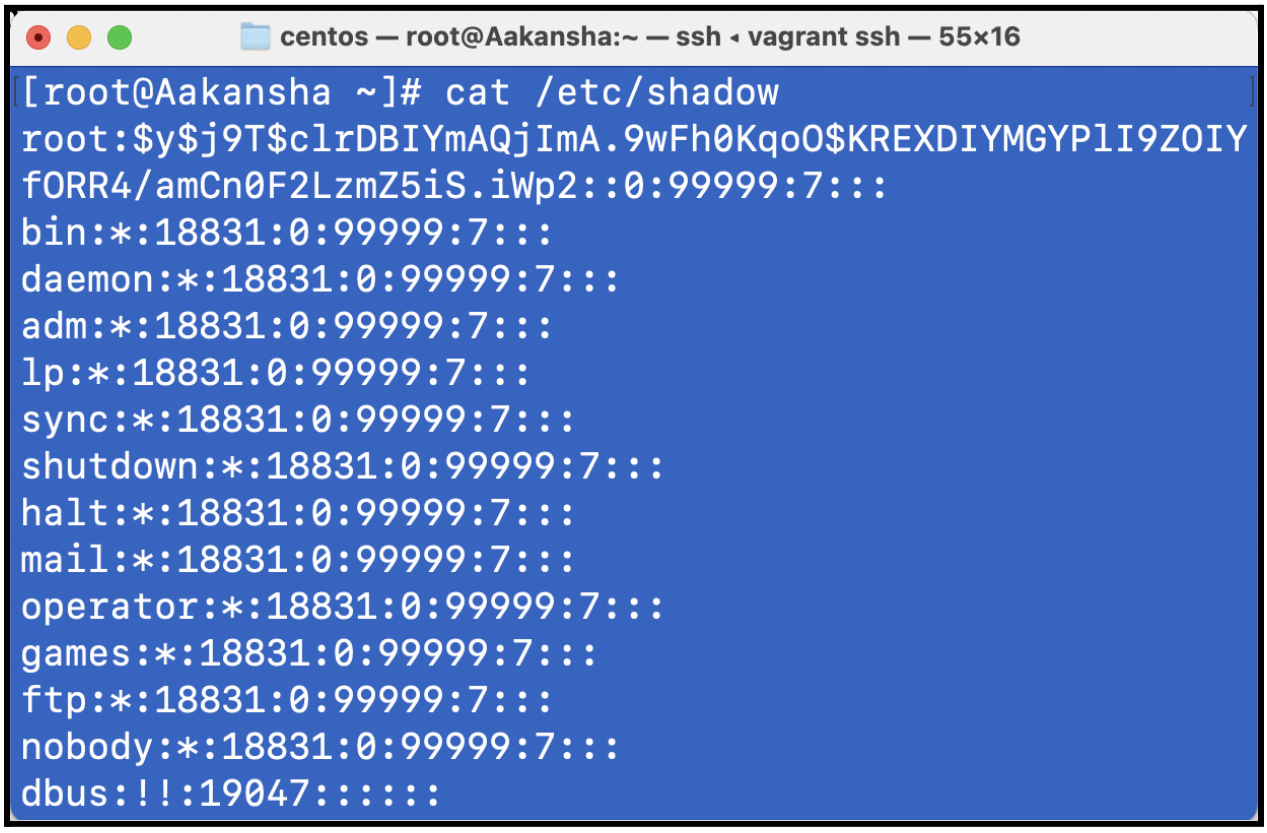
DELETE USER & GROUP



3. The /etc/shadow file

This file stores users' password and password related information. Just like /etc/passwd file, this file also uses an individual line for each entry.

A. Username
B. Encrypted password
C. Number of days when password was last changed
D. Number of days before password can be changed
E. Number of days after password must be changed
F. Number of days before password expiry date to display the warning message
G. Number of days to disable the account after the password expiry
H. Number of days since the account is disabled
I. Reserved field



USER & GROUP Cheatsheet

COMMANDS	DESCRIPTION
useradd	Creates user in RedHat
adduser	Creates user in ubuntu
id	Shows user info
groupadd	Creates group
Usermod -G groupname username	Adds user to group
passwd	Set / reset password
userdel -r	Removes user with home dir
groupdel	Removes group
last	Shows last login in system
who	Who is logged into system
whoami	Shows username
lsof -u user	Lists files opened by the user

6. File permissions

File permissions in Linux or Unix systems define who can read, write, or execute a file or directory. Each file or directory is assigned permissions for three categories of users: owner, group, and others.

Permission Types:

- **Read (r):** Allows reading the contents of the file/directory.
- **Write (w):** Allows modifying the contents of the file/directory.
- **Execute (x):** Allows executing a file or traversing a directory.

Permission Categories:

1. **Owner (u):** The user who owns the file.
2. **Group (g):** A group of users who are allowed to access the file.
3. **Others (o):** Everyone else.

File Permission Representation:

- Each file has 10 characters representing its permissions.
- The first character indicates the type of file (`-` for regular file, `d` for directory).
- The next nine characters are divided into three sets of three:
    1. Owner permissions (user)
    2. Group permissions
    3. Other permissions

Example:

```
-rwxr-xr--
```

- **rwx:** Owner can read, write, and execute.
- **r-x:** Group can read and execute, but not write.
- **r--:** Others can only read.

Changing File Permissions:

The `chmod` command is used to change the permissions of files and directories.

Symbolic Mode:

chmod u+x filename : Adds execute permission to the owner

chmod g-w filename : Removes write permission from the group

chmod o+r filename : Adds read permission to others

Numeric Mode: File permissions are represented using numbers:

- **r** = 4
- **w** = 2
- **x** = 1

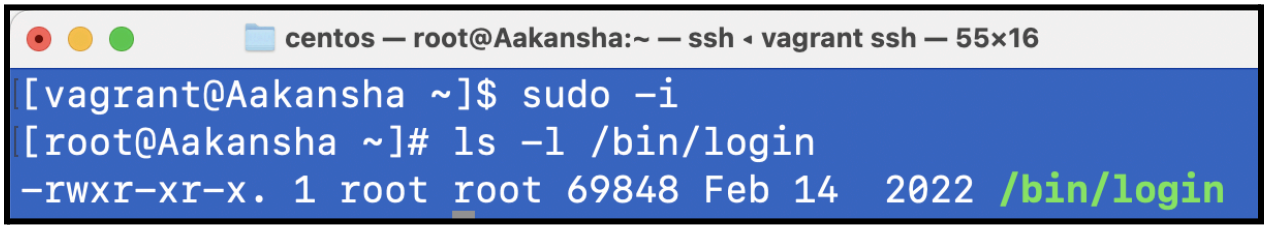
You add these numbers to set permissions:

- `chmod 755 filename` means:
  - **7** (Owner: `rwx` = 4 + 2 + 1)
  - **5** (Group: `r-x` = 4 + 1)
  - **5** (Others: `r-x` = 4 + 1)

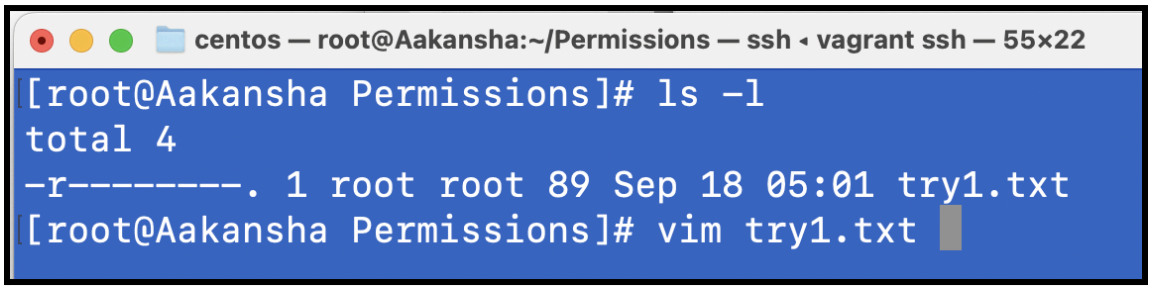
Example Permissions:

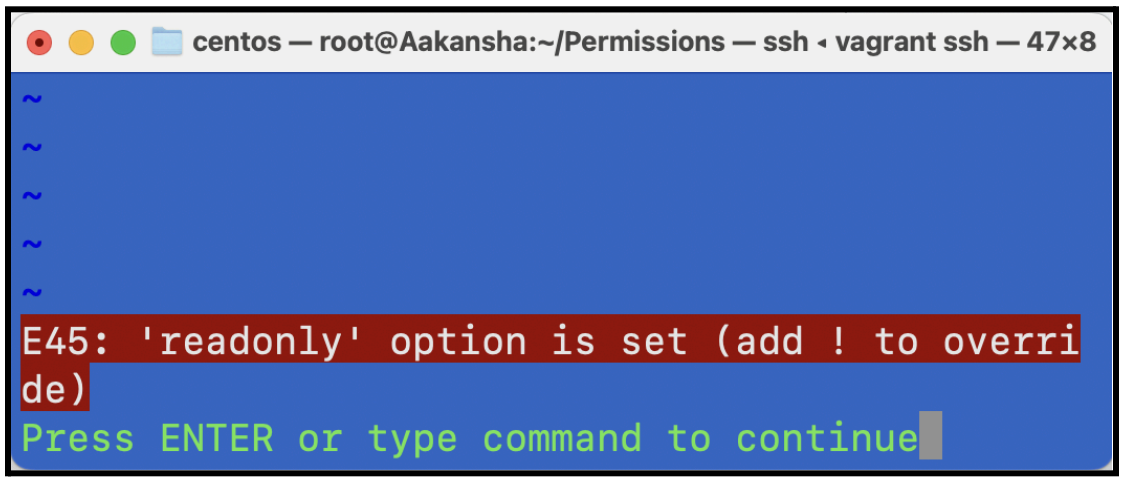
Command	Description
`chmod 777 file`	Full permissions (read, write, execute) to all.
`chmod 644 file`	Owner can read and write; group and others can only read.
`chmod u+x file`	Adds execute permission to the owner.
`chmod g-r file`	Removes read permission from the group.

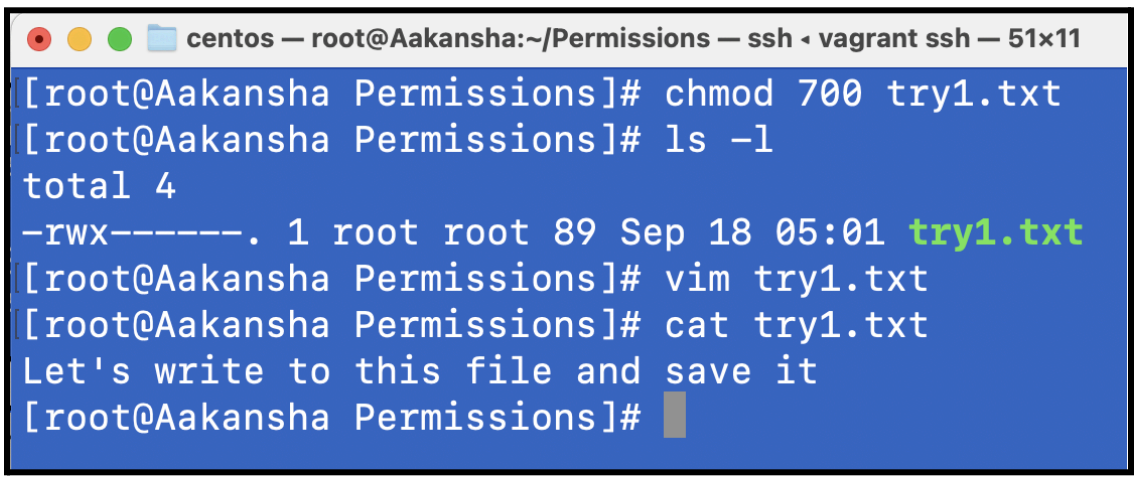
Viewing Permissions from the Command-Line



Changing File Permissions







7. Sudo

The sudo command in Linux allows you to run programs or commands with superuser (root) privileges. It stands for "superuser do," and it is commonly used when you need administrative permissions to perform certain tasks, like installing software, modifying system files, or changing system settings.

Example Uses of `sudo`:

1. **Installing software (requires root privileges):**

```
sudo apt-get install vim
```

2. **Editing system files (e.g., using `vim` or `nano`):**

```
sudo nano /etc/hosts
```

3. **Changing file permissions (e.g., making a script executable):**

```
sudo chmod +x script.sh
```

4. **Rebooting the system:**

```
sudo reboot
```

How to Use sudo:

- When you run a command with sudo, you'll usually be asked to enter your password. This is your regular user password, not the root password.
- After entering your password, the command will execute with elevated privileges.

Granting sudo Privileges:

If you don't have sudo permissions but need them, a system administrator can add you to the sudo group using:

```
sudo usermod -aG sudo yourusername
```