Dealing with Users, Groups, Permissions, and Processes

Session 3

Agenda

01

Dealing with users and groups

02

File permissions and ownership

03

Processes

01

Dealing with users and groups

Dealing with Users and Groups

1. What is a user and types of users
2. What is a group and types of groups
3. Create, switch and delete a user
4. Create and delete a group
5. Add a user to a group

What is the user?

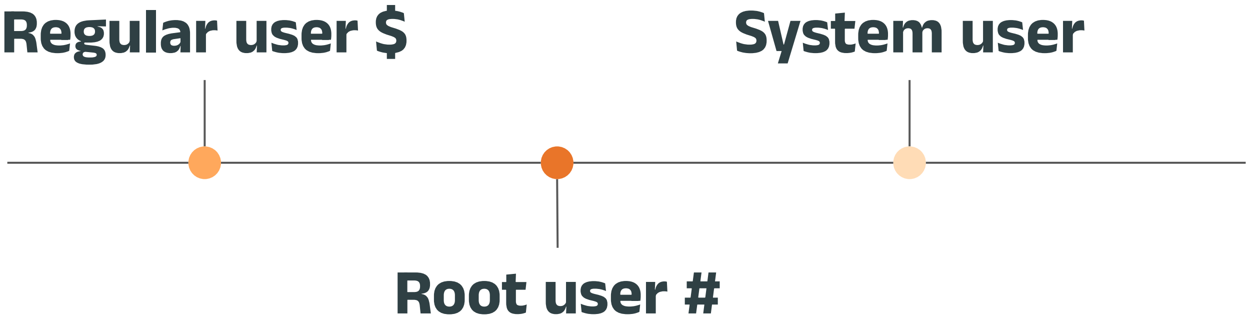
- A **login account** that allows someone to access and use the computer.Each user has their own **name, password, files, and permissions** .

- The system distinguishes user accounts by the **unique identification number** assigned to them, the **user ID** or **UID**

Why Are There Users in Linux?

- **Organization** :Everyone has their own workspace,Every file has a particular user as its owner.
- **Security** :Users can't access or break each other's files. Every process (running program) on the system runs as a particular user.
- **Control** : You can give different permissions to different users

Types of Users



Regular user $

Root user #

System user

1- Regular User $-> UID >= 1000

Used for daily tasks with limited permissions to keep the system secure and stable.

- Can only modify their own files.
- Can not modify system files or other users' files.

2- Superuser (root) # -> UID = 0

The Administrator of the system and has all permissions.

- Modify the system configurations
- Manage users and permissions
- Access or delete any file

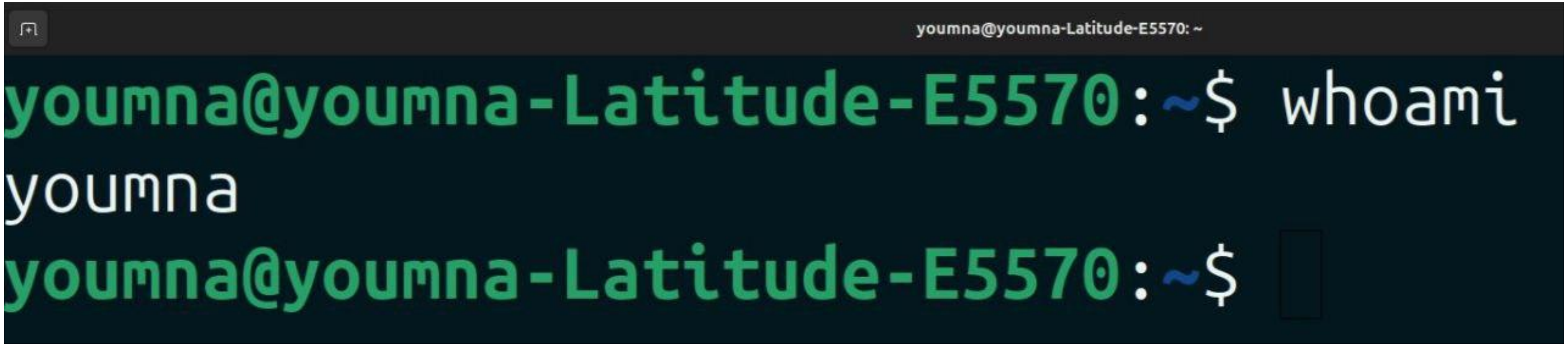
3- System User (Service Account) -> UID (1:999)

Created automatically by the system when installing a software to run background service.

- Security Isolation
- Track the actions of different services

To display The current user

→ **whoami**



```
youmna@youmna-Latitude-E5570:~$ whoami
youmna
youmna@youmna-Latitude-E5570:~$ 
```

Groups

Groups

- A group can contain **multiple users**: All users belonging to a group will have the **same permissions access to the files.**

- Groups have group names to make them easier to work with. Internally, the system distinguishes groups by **the unique identification number** assigned to them, the **group ID or GID.**

Types of Groups

- Primary Group
- Supplementary Groups

1- Primary Group

- A primary group is the default group assigned to a user when they are created on the system.
- By default, this group has the same name as the user.
- Any files created by the user will belong to their primary group unless specified otherwise.
- Each user can have only **one primary group**.

2- Supplementary Group

- Any other group a user belongs to other than the primary group.
- The user can be member of multiple secondary groups

Dealing with Users

Linux is a Multi-User Environment

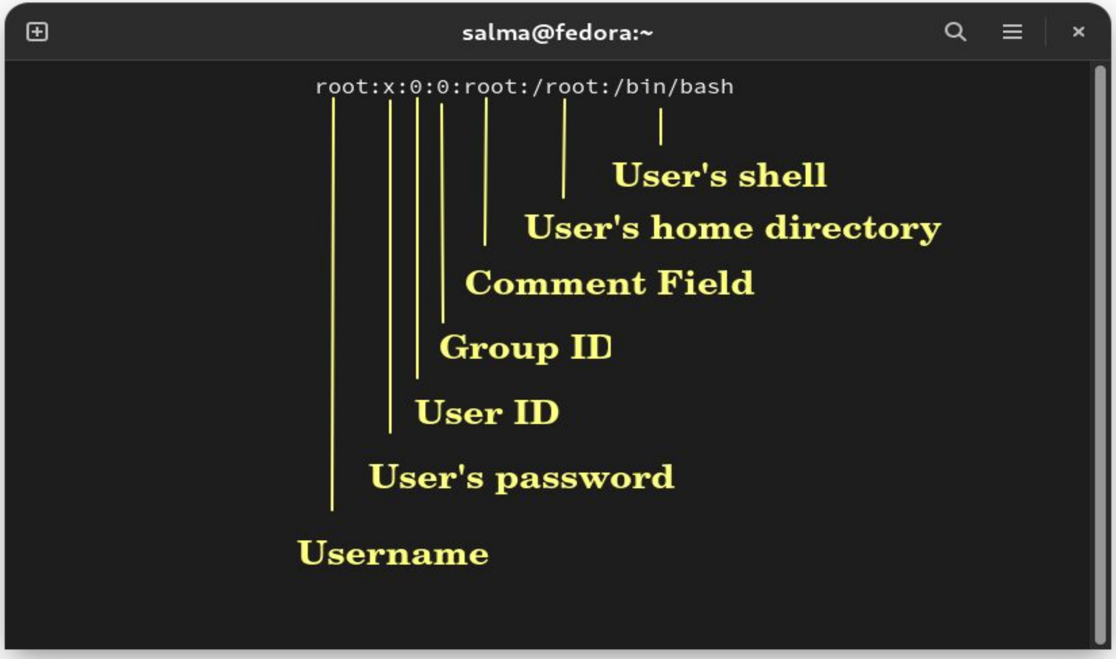
Linux is designed to allow **multiple users** to use the system **at the same time**

→ That means:

- Many users can be **logged in** .
- Each user has their **own files, settings, and permissions**.
- The system keeps **users isolated** from each other for security and privacy.

/etc/passwd File

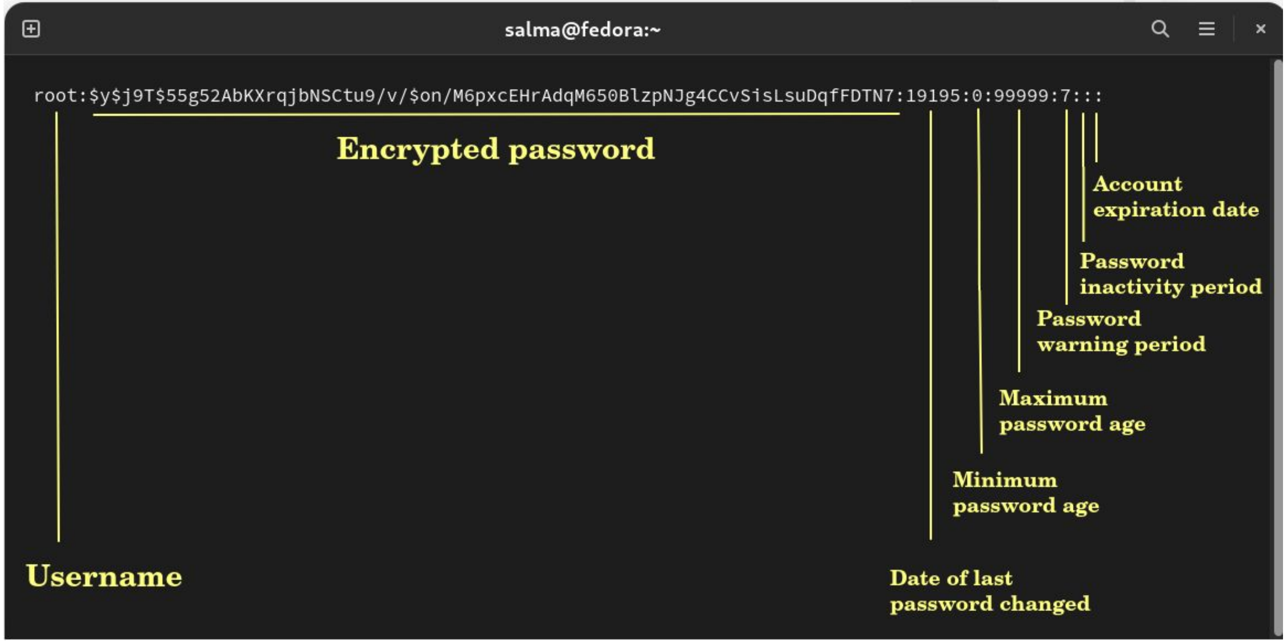
Stores information about the users on the system, each line is giving information about a different user



```
root:x:0:0:root:/root:/bin/bash
```

/etc/shadow File

Store information about user authentication. It requires superuser read permissions.



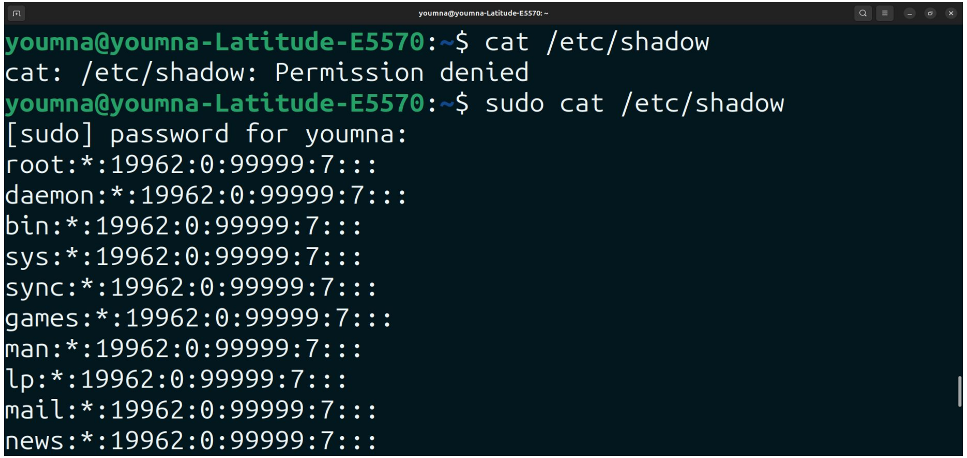
```
root:$y$j9T$55g52AbKXrqjbNSCtu9/v/$on/M6pxcEHrAdqM650BlzpNJg4CCvSisLsuDqfFDTN7:19195:0:99999:7:::
```

etc/shadow is a Protected File

This file contains sensitive information mainly **encrypted user passwords**.That’s why it’s only readable by the root use **(a file only accessed by the root user)**

So how can a regular user view it?

You need to temporarily borrow **root’s power** using **sudo**:



```
youmna@youmna-Latitude-E5570:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
youmna@youmna-Latitude-E5570:~$ sudo cat /etc/shadow
[sudo] password for youmna:
root:*:19962:0:99999:7:::
daemon:*:19962:0:99999:7:::
bin:*:19962:0:99999:7:::
sys:*:19962:0:99999:7:::
sync:*:19962:0:99999:7:::
games:*:19962:0:99999:7:::
man:*:19962:0:99999:7:::
lp:*:19962:0:99999:7:::
mail:*:19962:0:99999:7:::
news:*:19962:0:99999:7:::
```

Sudo → SuperUser Do

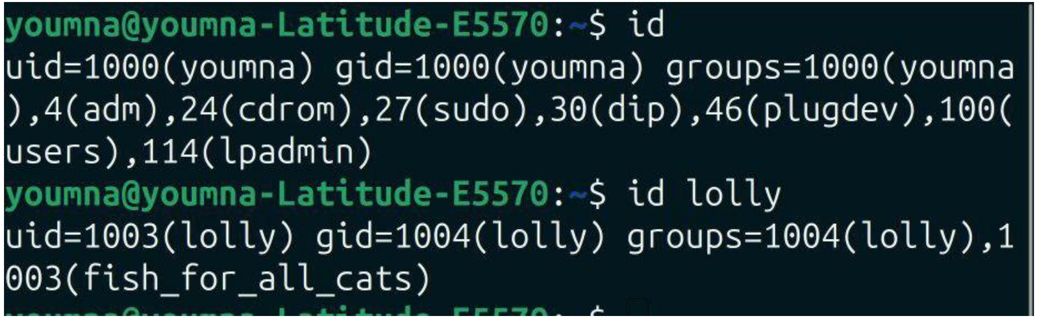
☆ It’s a Linux command that allows a regular user to execute commands with **root (superuser) permissions** temporarily, Instead of logging in as the root user

☆ **Access Control** :Only users in the **sudo group** (`sudoers`) can use `it`.

☆ So, if a user is **not** in the `this group`, they will get a **“Permission denied”** or **“is not in the sudoers file”** message when trying to use it.

To display The current user

→ **Id command**



```
youmna@youmna-Latitude-E5570:~$ id
uid=1000(youmna) gid=1000(youmna) groups=1000(youmna
),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),100(
users),114(lpadmin)
youmna@youmna-Latitude-E5570:~$ id lolly
uid=1003(lolly) gid=1004(lolly) groups=1004(lolly),1
003(fish_for_all_cats)
youmna@youmna-Latitude-E5570:~$
```

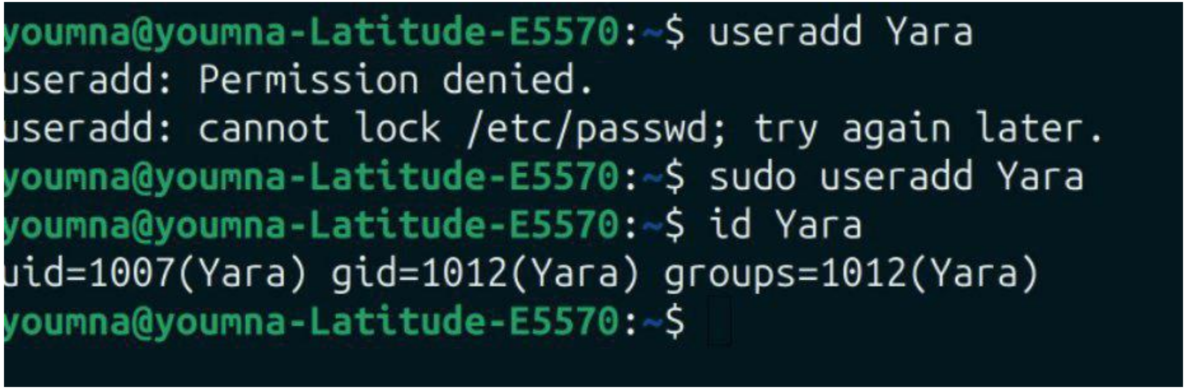
→ **Id <username>**: Display id for selected user

Creating User

Only the **root user** or someone with **sudo privileges** can create new users.

⟶Using command : **useradd**

Syntax: sudo useradd [options] <username>



```
youmna@youmna-Latitude-E5570:~$ useradd Yara
useradd: Permission denied.
useradd: cannot lock /etc/passwd; try again later.
youmna@youmna-Latitude-E5570:~$ sudo useradd Yara
youmna@youmna-Latitude-E5570:~$ id Yara
uid=1007(Yara) gid=1012(Yara) groups=1012(Yara)
youmna@youmna-Latitude-E5570:~$
```

Common options with command useradd

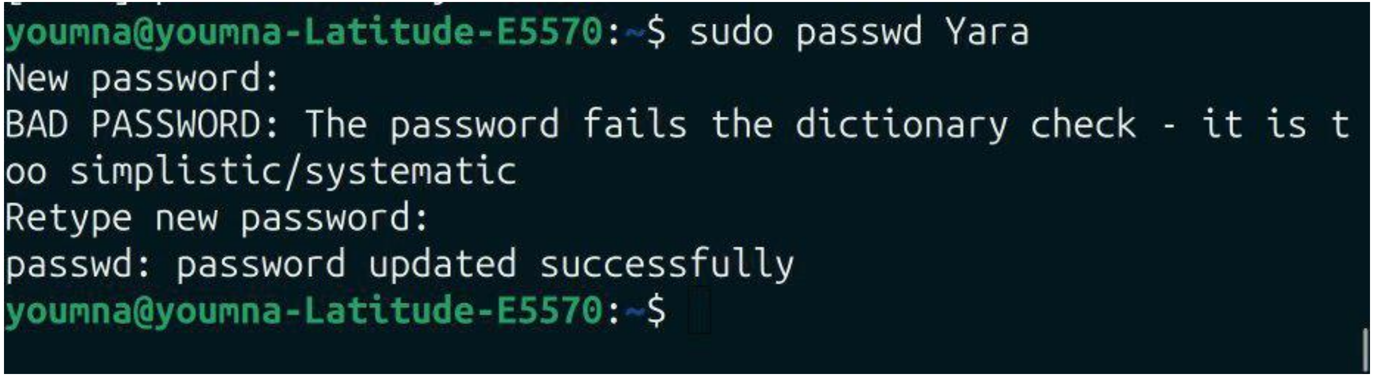
Option	Meaning	Example
-m	Automatically creates `/home/sara`	useradd -m sara
-G	Adds user to extra groups e.g (supplementary groups)	useradd -G game,video sara

Setting/Changing Passwords

Using command **passwd**

Syntax: sudo passwd <username>

The **root user or a user with sudo privilege** to change or set passwords for other accounts.



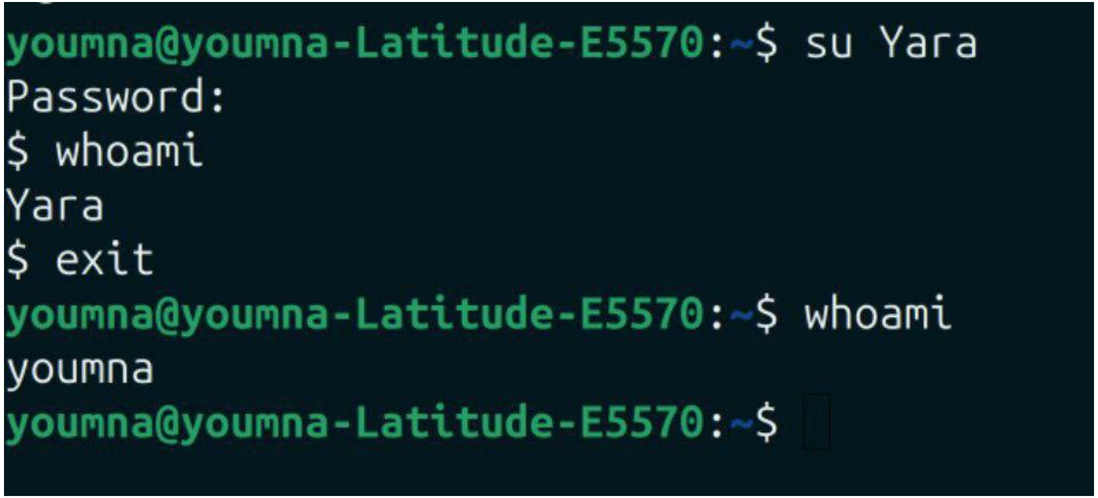
```
youmna@youmna-Latitude-E5570:~$ sudo passwd Yara
New password:
BAD PASSWORD: The password fails the dictionary check - it is t
oo simplistic/systematic
Retype new password:
passwd: password updated successfully
youmna@youmna-Latitude-E5570:~$
```

Switching users

Using command **su** -> [switch user]

Syntax: su <username>

It asks for the password of the user you want to switch to

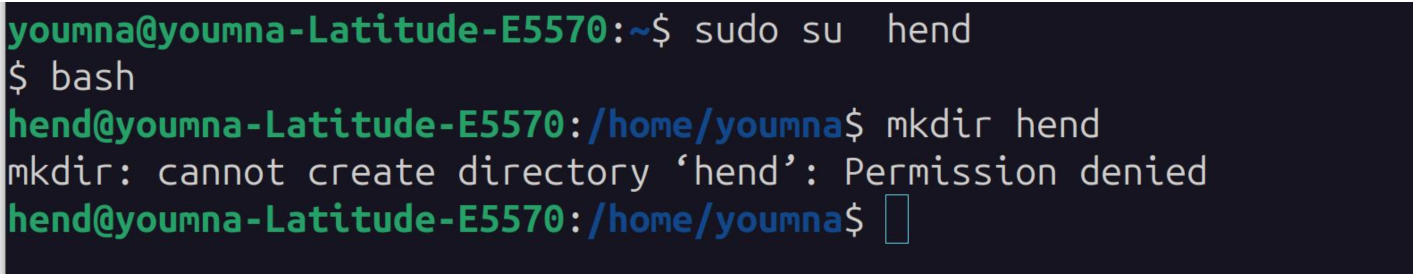


```
youmna@youmna-Latitude-E5570:~$ su Yara
Password:
$ whoami
Yara
$ exit
youmna@youmna-Latitude-E5570:~$ whoami
youmna
youmna@youmna-Latitude-E5570:~$
```

Switching users

sudo su <username>

It asks for the current user's password, but it inherits the current working directory from the previous user



```
youmna@youmna-Latitude-E5570:~$ sudo su  hend
$ bash
hend@youmna-Latitude-E5570:/home/youmna$ mkdir hend
mkdir: cannot create directory ‘hend’: Permission denied
hend@youmna-Latitude-E5570:/home/youmna$ 
```

How to fix it?

Switching users

```
sudo su - [username]
```

This command starts a **login shell** for this user , which:

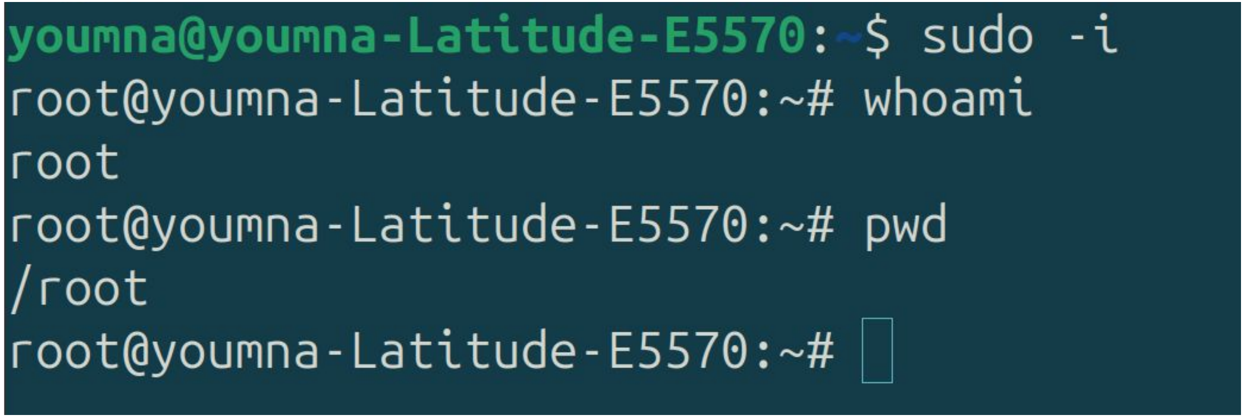
- Loads the user’s environment variables
- Starts in their home directory: `/home/username`
- `‘warning: cannot change directory to /home/username: No such file or directory’`

`you will get this warning if you switch to user does not have home directory`

Switching users

Sudo -i

It switches to root user , Loads **root’s full environment** (including files from root’s home).



```
youmna@youmna-Latitude-E5570:~$ sudo -i
root@youmna-Latitude-E5570:~# whoami
root
root@youmna-Latitude-E5570:~# pwd
/root
root@youmna-Latitude-E5570:~# 
```

Deleting User

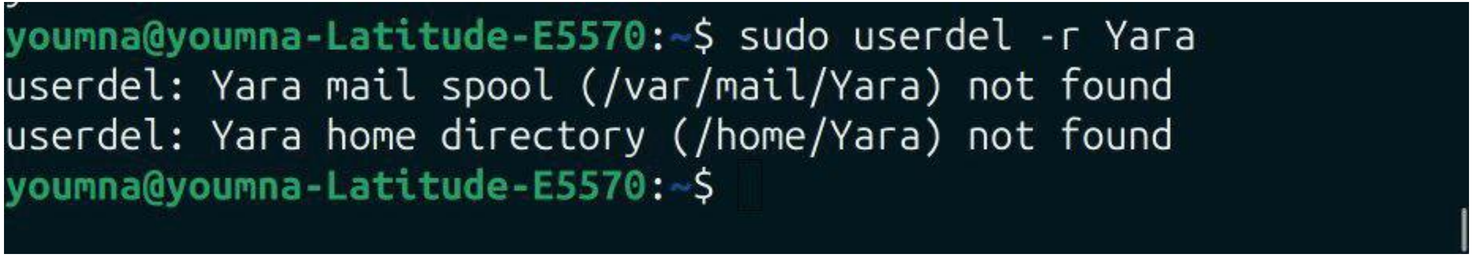
userdel is used to remove the details of username from /etc/passwd without
removing the user's home directory by default.

Syntax: sudo uesrdel [options] <username>

→ common option : **-r**

Deleting User

- When -r flag is specified, the userdel command also removes the user's home directory.
- If you don’t use -r, the user’s files will still exist and be owned by their **old UID**, That could lead to **security risks** or **data leaks** later.

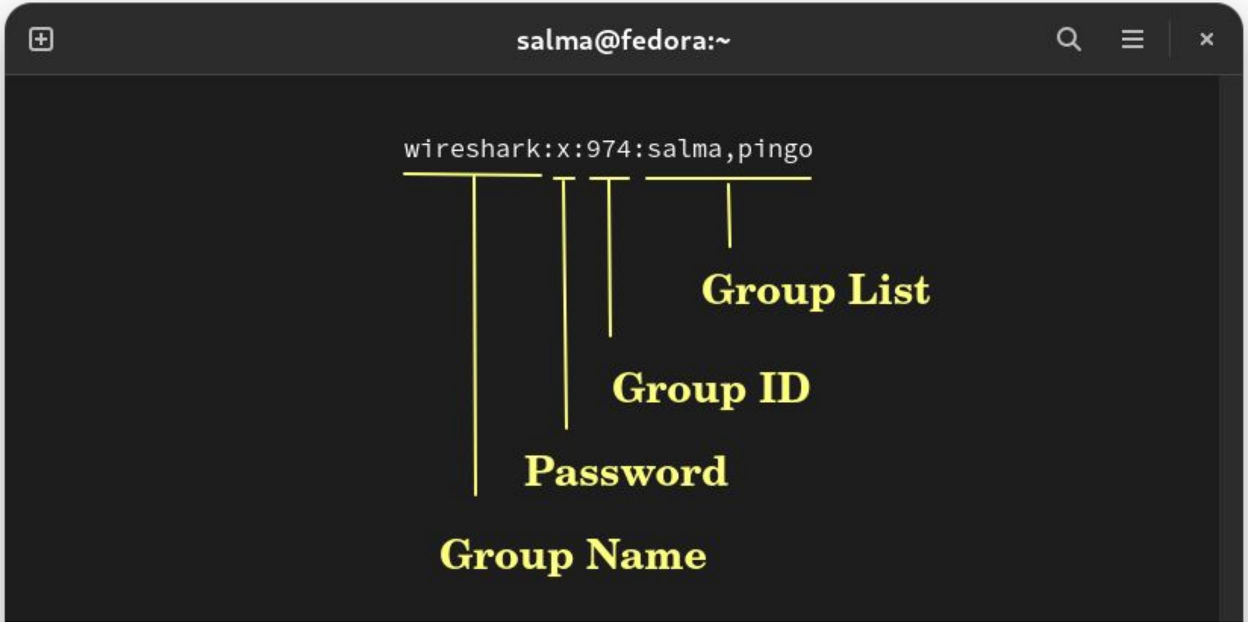


```
youmna@youmna-Latitude-E5570:~$ sudo userdel -r Yara
userdel: Yara mail spool (/var/mail/Yara) not found
userdel: Yara home directory (/home/Yara) not found
youmna@youmna-Latitude-E5570:~$
```

Dealing with Groups

Dealing With Groups

- **/etc/group** file stores information about all groups in the system



```
wireshark:x:974:salma,pingo
```

Dealing With Groups

→ **groups**

list all groups you are a member of

→ **groups <username>**

list all groups of a specific user

→ **cat /etc/group**

List all groups for all users not only for the current user

Creating a Group

→ **Syntax** : groupadd Group_name

→ Only the root or a user with sudo privileges can create new groups.

Adding a User to a Group

- Existing users accounts are added to groups using the **usermod** command.

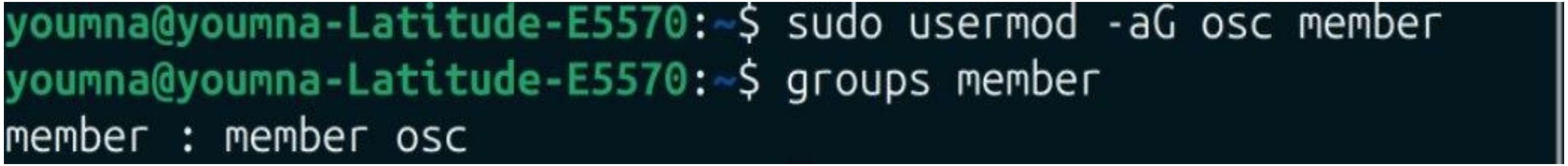
- **syntax** : sudo usermod [options] <group name> <username>

- The -G option tells the command that we will add the user to a supplementary group . The -a option puts the command in append mode ; otherwise , the command will remove the user from all groups unspecified in the command.

Example

To add the user “member” to the group “osc“ we will write

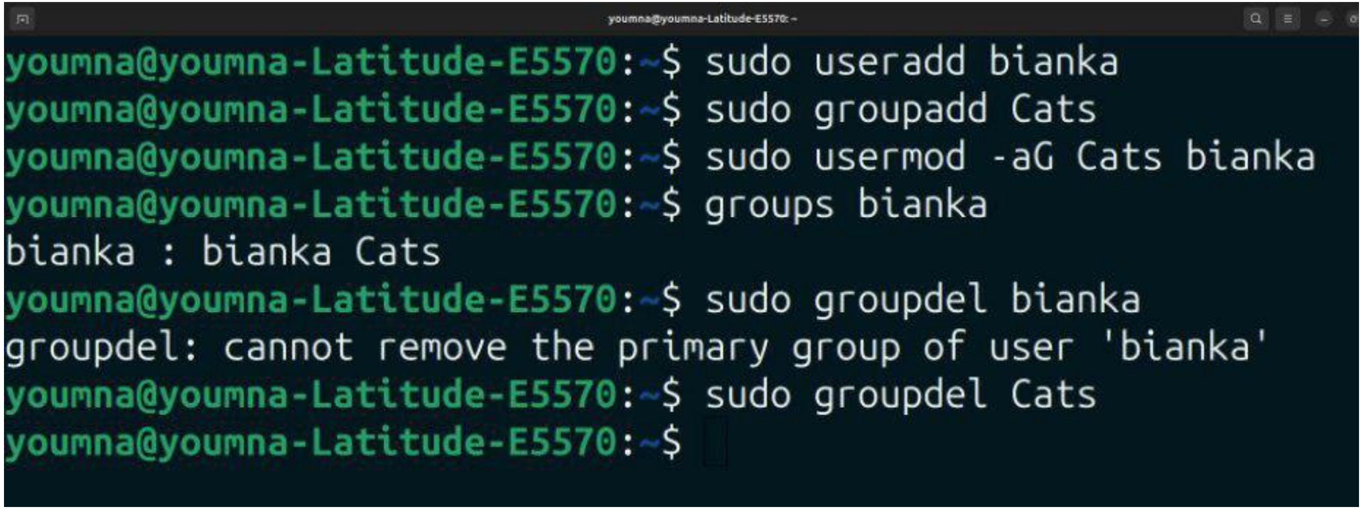
usermod -aG osc member



```
youmna@youmna-Latitude-E5570:~$ sudo usermod -aG osc member
youmna@youmna-Latitude-E5570:~$ groups member
member : member osc
```

Deleting groups

- Groups are deleted using the groupdel command
- **syntax** : groupdel <group name>
- You cannot delete the primary group of a user account



```
youmna@youmna-Latitude-E5570:~$ sudo useradd bianka
youmna@youmna-Latitude-E5570:~$ sudo groupadd Cats
youmna@youmna-Latitude-E5570:~$ sudo usermod -aG Cats bianka
youmna@youmna-Latitude-E5570:~$ groups bianka
bianka : bianka Cats
youmna@youmna-Latitude-E5570:~$ sudo groupdel bianka
groupdel: cannot remove the primary group of user 'bianka'
youmna@youmna-Latitude-E5570:~$ sudo groupdel Cats
youmna@youmna-Latitude-E5570:~$
```

Whoami-Game ?!

1- I’m the administrator in any Linux system, My UID is always ‘O’.

If I disappear, the whole system might cry

2- Ask me and I’ll tell you who you are

3- If you no longer want a user on your system… Call me and I’ll make them disappear.

4- I grant you temporary superpowers. Just say the magic word [me]

but only if you're on the VIP list.

5- When you want to form a club of users , I’m the one who makes that group official.

02

File Permissions and ownership

What if someone else could read/delete your private documents without you knowing?

That’s why understanding permissions and ownership

File Permissions and ownership

1. File Ownership
2. File Permissions
3. Change file ownership and group
4. Change file Permissions

Linux File Ownership

Every file and directory on your Unix/Linux system is assigned **3 types of owner:**

1-User (Owner)

A user is the owner of the file. By default, the person who created a file becomes its owner

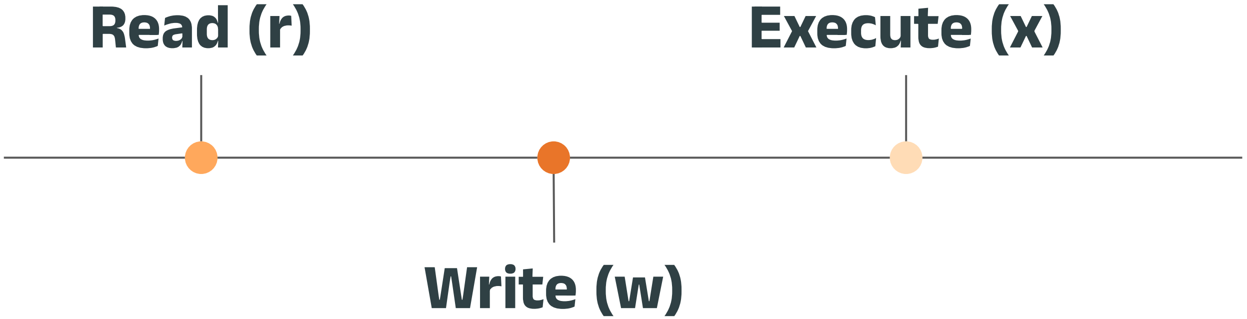
2-Group

A group can contain multiple users. All users belonging to a group will have the same Linux group permissions access to the file.

3-Other

Any other user who has access to a file. This person has neither created the file, nor he belongs to a group who could own the file. Practically, it means everybody else. Hence, when you set the permission for others.

Every file or directory on Unix/Linux system has 3 possible permissions:



Read (r)

Write (w)

Execute (x)

1- Read (r) Permission

File: gives you the authority to open and read a file.

→ cat file_name

Directory: gives you the ability to list its Content.

→ ls dir

2- Write (w) Permission

File: gives you the authority to modify the contents of a file.

→ nano file

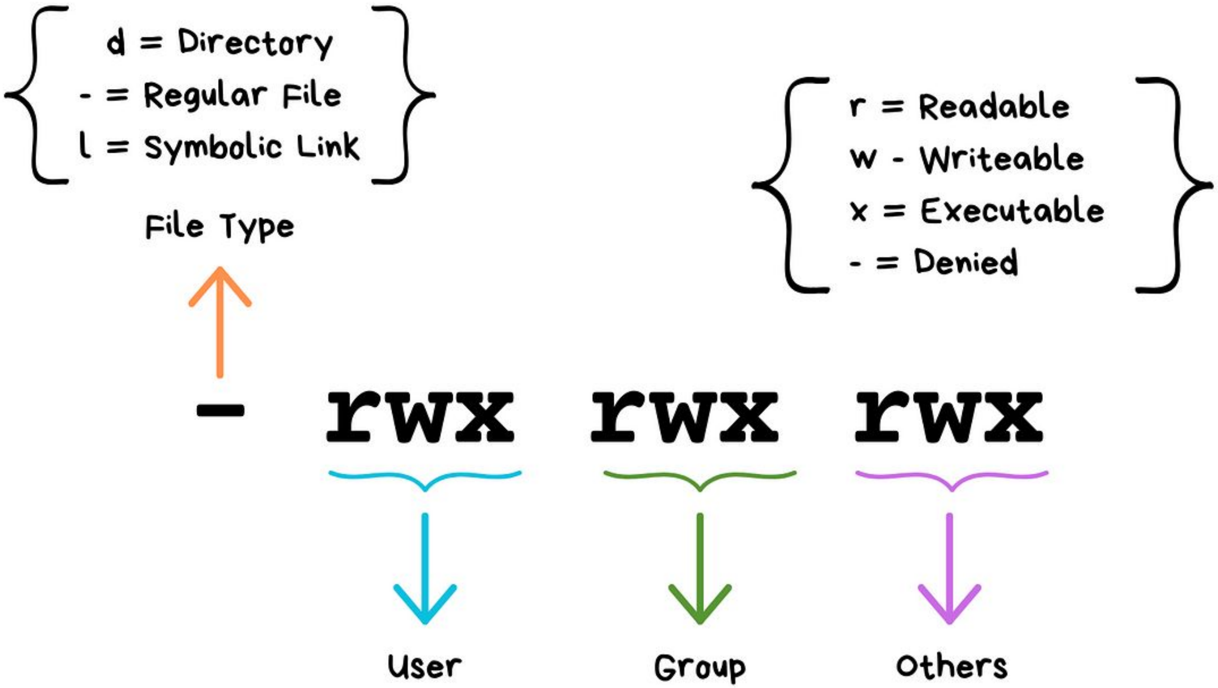
Directory: gives you the authority to add, remove, and rename files stored in the directory.

→ touch/rm file in dir

3- Execute (x) Permission

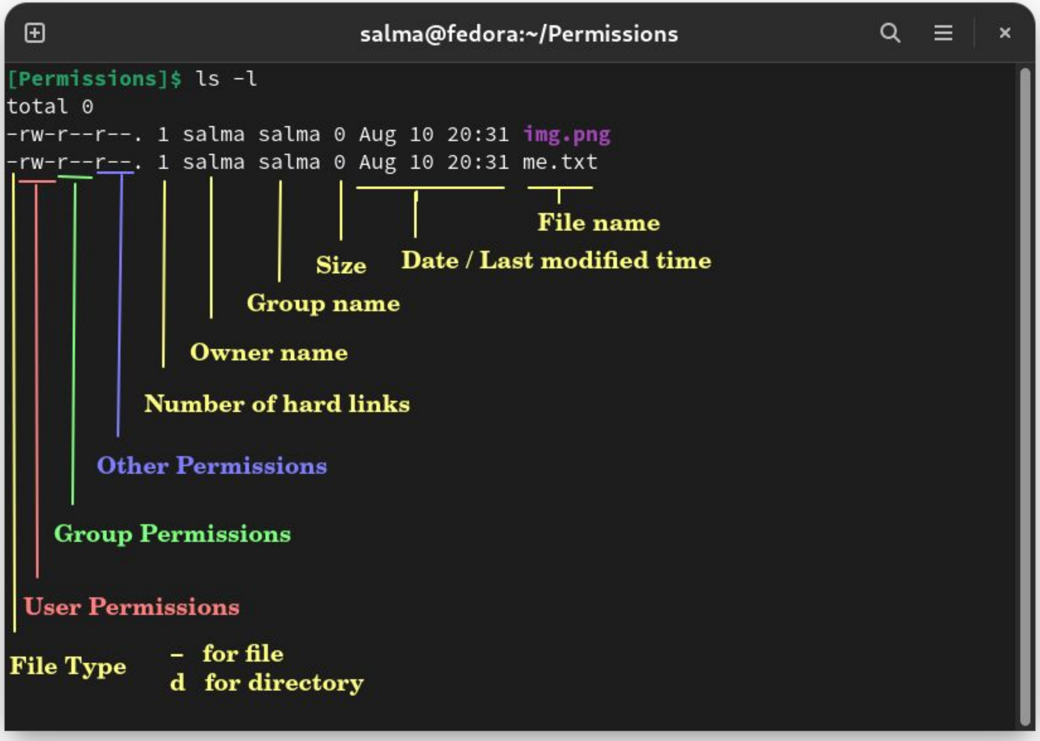
File: In Unix/Linux, you cannot run a program unless the execute permission is set.
By default, any newly created files are not executable regardless of their file extension suffix.
→ run script
Directory: The contents of the directory can be accessed.
→ cd dir

Linux File Ownership



Display permissions:

Using command : **ls -l <file/dir name>**

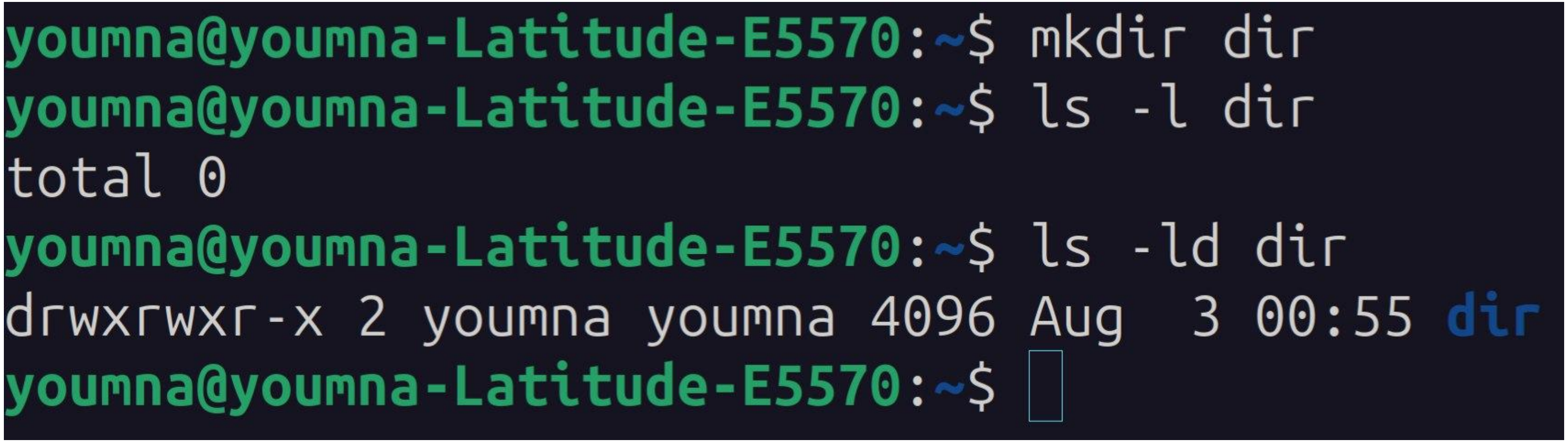


```
[Permissions]$ ls -l
total 0
-rw-r--r--. 1 salma salma 0 Aug 10 20:31 img.png
-rw-r--r--. 1 salma salma 0 Aug 10 20:31 me.txt
```

Display Permissions

Use option -d For empty directory

Syntax : ls -ld [dir_name]



```
youmna@youmna-Latitude-E5570:~$ mkdir dir
youmna@youmna-Latitude-E5570:~$ ls -l dir
total 0
youmna@youmna-Latitude-E5570:~$ ls -ld dir
drwxrwxr-x 2 youmna youmna 4096 Aug  3 00:55 dir
youmna@youmna-Latitude-E5570:~$ 
```

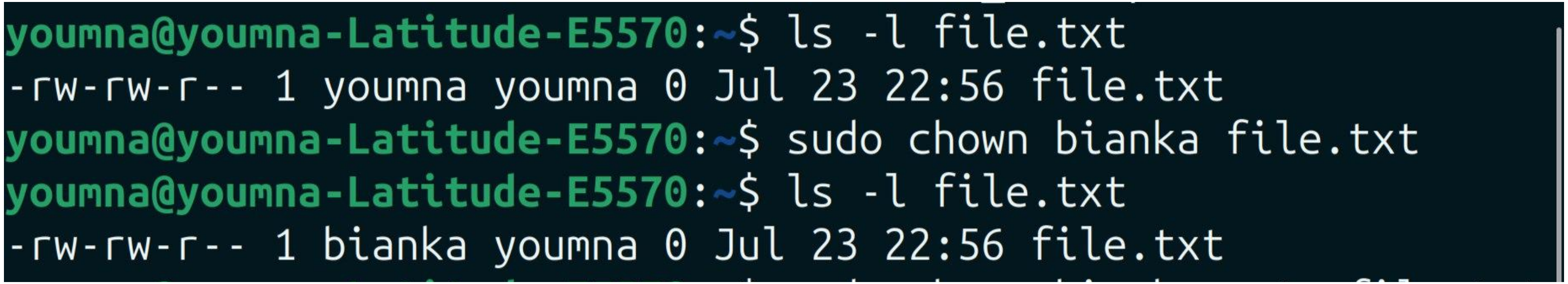
Change File Ownership and Group

Changing ownership:

⟶ for changing the ownership of a file/directory, you can use : chown user filename

⟶ Chown :[change owner]

⟶ **Syntax**: sudo chown [new_username] [file name]



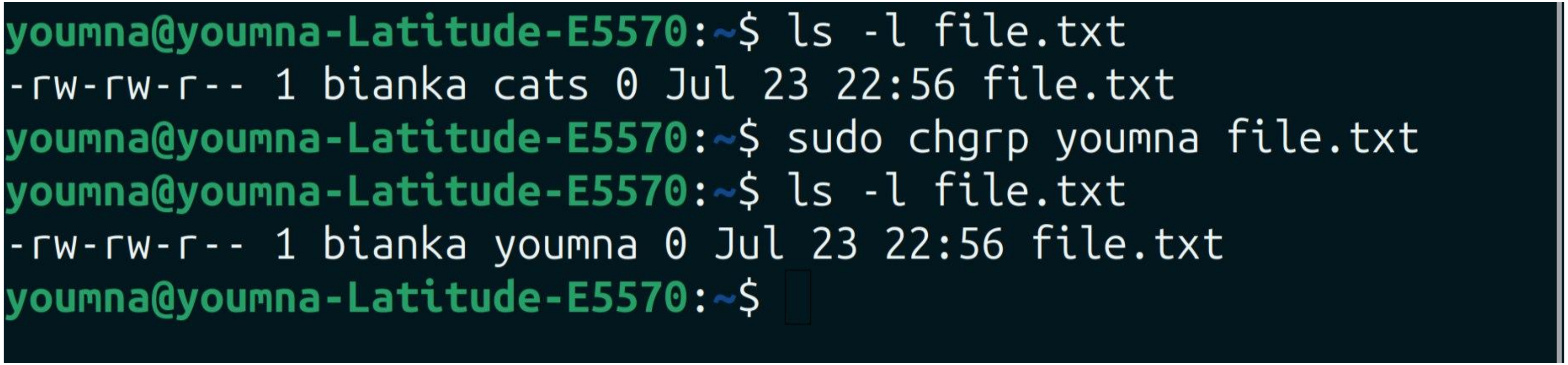
```
youmna@youmna-Latitude-E5570:~$ ls -l file.txt
-rw-rw-r-- 1 youmna youmna 0 Jul 23 22:56 file.txt
youmna@youmna-Latitude-E5570:~$ sudo chown bianka file.txt
youmna@youmna-Latitude-E5570:~$ ls -l file.txt
-rw-rw-r-- 1 bianka youmna 0 Jul 23 22:56 file.txt
```

Changing group

→ To change group-owner : **chgrp group_name filename**

→ Chgrp :[change group]

→ **Syntax**: sudo chgrp [new_group] [file name]



```
youmna@youmna-Latitude-E5570:~$ ls -l file.txt
-rw-rw-r-- 1 bianka cats 0 Jul 23 22:56 file.txt
youmna@youmna-Latitude-E5570:~$ sudo chgrp youmna file.txt
youmna@youmna-Latitude-E5570:~$ ls -l file.txt
-rw-rw-r-- 1 bianka youmna 0 Jul 23 22:56 file.txt
youmna@youmna-Latitude-E5570:~$ 
```

Change File Permission

Changing permissions

The chmod [change mode] command is used to change file/ directory's permissions, There are two ways:

1- **Symbolic mode**

2- **Absolute mode**

1- Symbolic mode

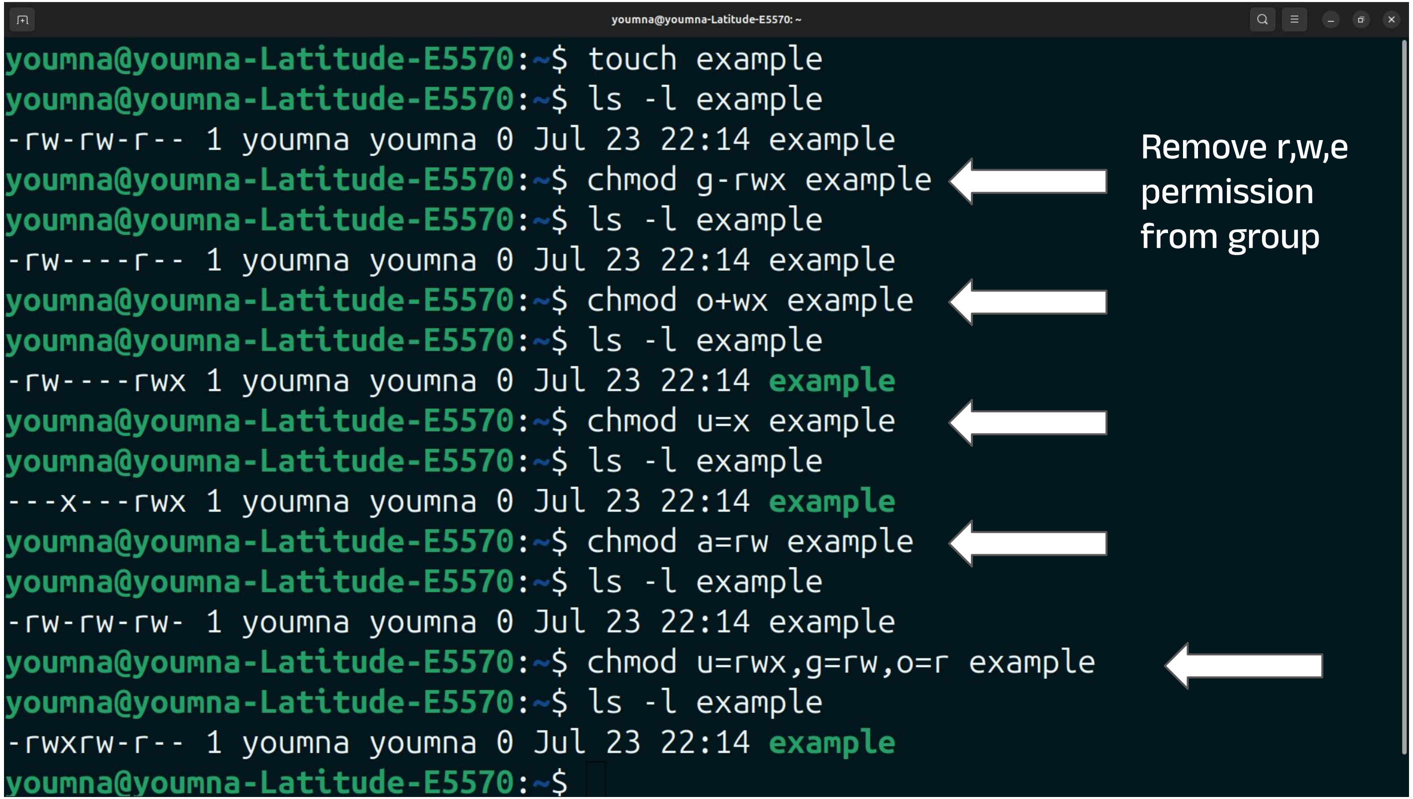
In symbolic mode, you can modify the permissions of a specific owner.

Syntax: chmod [ownerType] [operator] [new permission] [file name]

Owner Type	Symbol
User	**u**
Group	**g**
Other	**o**
All	**a**

Action	Operator
Add permission	**+**
Remove permission	**-**
Set exact permission	**=**

1- Symbolic mode Example

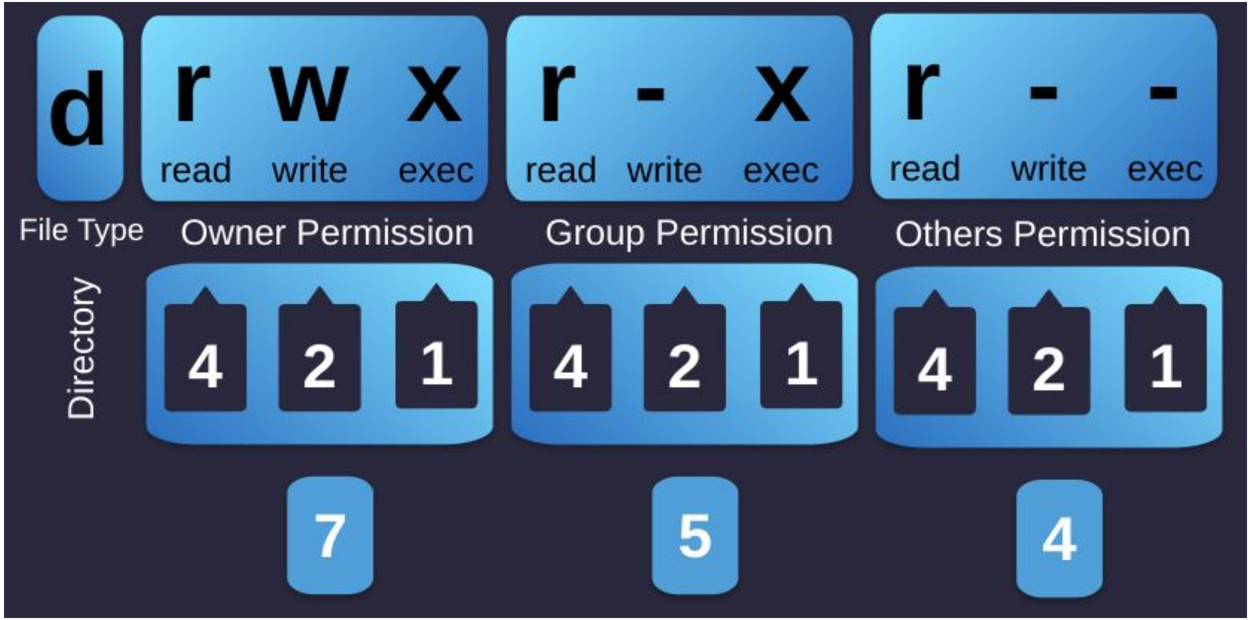


```
youmna@youmna-Latitude-E5570:~$ touch example
youmna@youmna-Latitude-E5570:~$ ls -l example
-rw-rw-r-- 1 youmna youmna 0 Jul 23 22:14 example
youmna@youmna-Latitude-E5570:~$ chmod g-rwx example
youmna@youmna-Latitude-E5570:~$ ls -l example
-rw----r-- 1 youmna youmna 0 Jul 23 22:14 example
youmna@youmna-Latitude-E5570:~$ chmod o+wx example
youmna@youmna-Latitude-E5570:~$ ls -l example
-rw----rwx 1 youmna youmna 0 Jul 23 22:14 example
youmna@youmna-Latitude-E5570:~$ chmod u=x example
youmna@youmna-Latitude-E5570:~$ ls -l example
---x---rwx 1 youmna youmna 0 Jul 23 22:14 example
youmna@youmna-Latitude-E5570:~$ chmod a=rw example
youmna@youmna-Latitude-E5570:~$ ls -l example
-rw-rw-rw- 1 youmna youmna 0 Jul 23 22:14 example
youmna@youmna-Latitude-E5570:~$ chmod u=rwx,g=rw,o=r example
youmna@youmna-Latitude-E5570:~$ ls -l example
-rwxrw-r-- 1 youmna youmna 0 Jul 23 22:14 example
youmna@youmna-Latitude-E5570:~$
```

Remove r,w,e permission from group

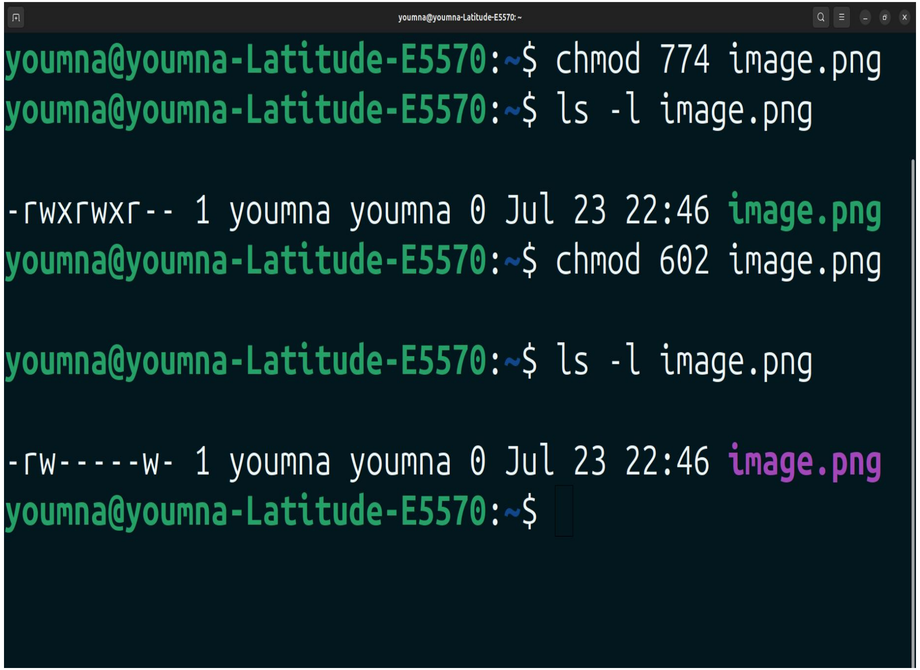
2- Absolute mode

- We Use **Binary System** in File Permissions:
  Because each permission is either on (enabled) or off (disabled)



2- Absolute mode

Permission	Symbol	Num.
No permissions	— — —	0
Execute only	— — x	1
Write only	— w —	2
Write and execute	— w x	3
Read only	r — —	4
Read and execute	r — x	5
Read and write	r w —	6
Read, write, execute	r w x	7



```
youmna@youmna-Latitude-E5570:~$ chmod 774 image.png
youmna@youmna-Latitude-E5570:~$ ls -l image.png

-rwxrwxr-- 1 youmna youmna 0 Jul 23 22:46 image.png
youmna@youmna-Latitude-E5570:~$ chmod 602 image.png

youmna@youmna-Latitude-E5570:~$ ls -l image.png

-rw-----w- 1 youmna youmna 0 Jul 23 22:46 image.png
youmna@youmna-Latitude-E5570:~$ 
```

Hands on #1

1- Create a user called student .

2- Create a group named interns and add student user to the group.

3-List groups of the student user.

4- Create a file called project_file , set its permissions, then list it:

- Owner : full access
- Group : read, write
- Others: no access

5- Change Ownership of project_file

- Owner : student
- Group : interns

6- **Finally**, remove user , group and file

Solution:

Step 1:

```
sudo useradd student
```

Step 2:

```
sudo groupadd interns
sudo usermod -aG interns student
```

Step 3:

```
groups student
```

Step 4:

```
touch project_file
chmod 760 project_file
ls -l project_file
```

Step 5:

```
sudo chown student:interns project_dir
```

Step 6:

```
sudo userdel -r student
sudo groupdel interns
rm  project_file
```

Break

03

Processes

Processes

1. Boot Process
2. Processes Definition
3. Process VS Job
4. Process Relationships
5. Process Management

Boot Process

What Is the Boot Process?

- The boot process is like your computer’s morning routine.

- the steps it takes to wake up and get ready for work.

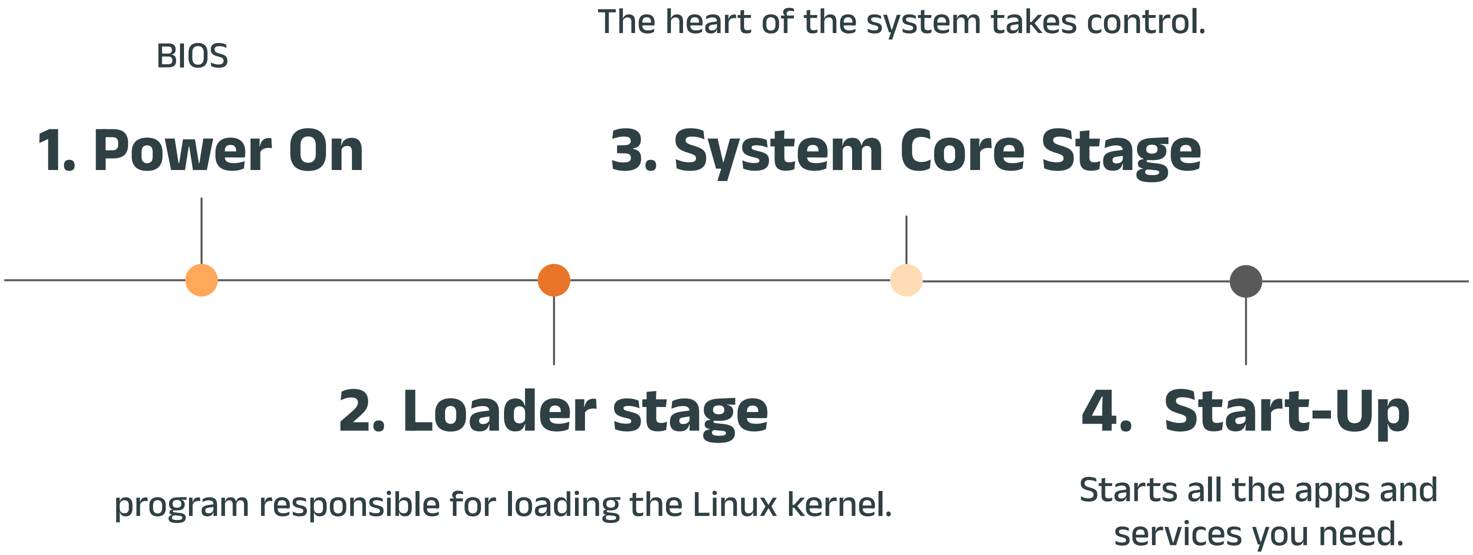
From pressing the power button -> to seeing your desktop screen.

Why Should You Care?

- Ever wondered what happens after you press the power button?
- Why does your computer take a while to start?
- What do those logos and black screens mean?

So let's see what is the Computers morning routine steps!

Boot Process Stages



BIOS

1. Power On

2. Loader stage

program responsible for loading the Linux kernel.

The heart of the system takes control.

3. System Core Stage

4. Start-Up

Starts all the apps and services you need.

Step 1: Power On

"Wake Up!"

- You press the power button
- Electricity flows into the machine
- The hardware gets powered up Just like you opening your eyes in the morning.

Step 1: Power On

BIOS/UEFI

- Is a Firmware on a chip (ROM) on motherboard.
- Perform the POST (power on self test) :

→ The computer checks its own hardware.

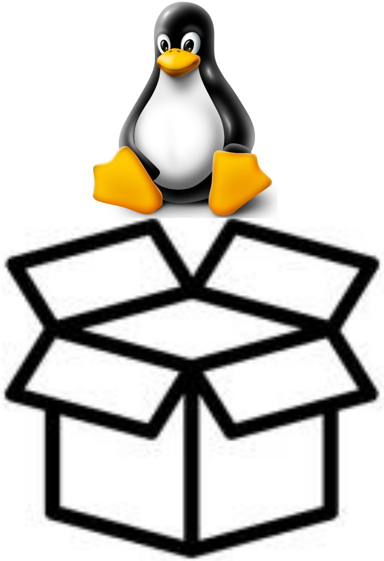
→ If something's wrong, you may hear beeps or see error messages.

8 beep error
&
blank display

Step 2: Loader stage

Finding the System

- A small program called the loader looks for the operating system (Linux).
- It prepares everything needed to start it.

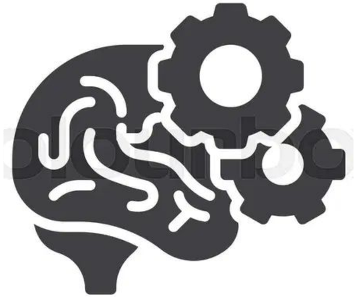


Step 3 : System Core (Kernel)

The Brain Takes Over

- The system’s core (called the Kernel) starts running.

- It controls everything in your computer:
  - ★ Memory
  - ★ CPU
  - ★ Devices
  - ★ etc

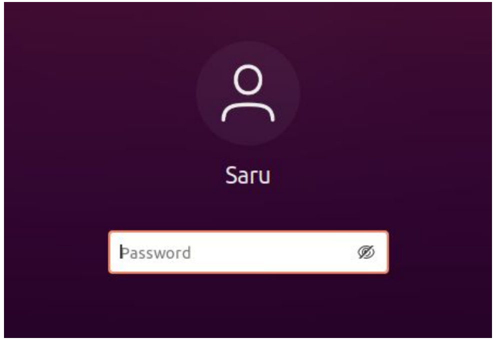


Step 4 : Start-Up (Init/Systemd)

Start-Up (Init/Systemd)

The system now starts all the things you see:

- The desktop
- Apps
- Background services



Saru

It’s like getting your room ready after waking up.

Finally the boot process ended now you see your login screen you can use your computer!

Processes Definition

Processes

- A process (or task) is an instance of a program.
- Each process has its own memory space and resources allocated to it.
- Each process has the illusion that it is the only process on the computer, but in reality all processes share system resources like the CPU and memory.

Processes

- A bash shell itself is a process, and running a script or program in it starts a new process.
- Each process in Linux has a process ID (PID) and is associated with a specific user and group.
- Linux is a multitasking operating system, which means that multiple processes can run at the same time.
- The kernel uses a scheduler to manage how different programs share the CPU.

Commands

Processes

ps

Lists processes associated with the current terminal session.

ps -e

prints all processes within the system.

ps -f

prints a more detailed output.

ps -u [user name]

prints a more detailed output.

ps -C [process name]

searches for a particular process name.

Processes

```
top
```

Shows a continuously updating display of the system processes with more information.

```
pstree
```

Display processes in a tree format.

```
ps aux --sort=-%mem
```

Sort processes based on memory usage

```
ps aux --sort=-%cpu
```

Sort processes based on CPU usage

Jobs

Jobs

- → A job is a concept used by the shell.
- → It is a unit of work performed by the user, which may consist of one or more processes.
- → For example, `ls | head` is a job consisting of two processes.

Jobs

- A job is identified by a job ID (JID).
- You can list the jobs of the current shell by running `jobs`

Processes VS Job

Process VS Jobs

- We will not go deep into the differences between processes and jobs, but just note that they are distinct but related concepts, and that some commands take PID while others take JID.

Foreground Processes

Foreground Processes

- These are initialized and controlled through a terminal session.
- There must be a user connected to the system to start such processes, as they don't start automatically as part of the system functions/services.
- Examples are user programs like web browsers, image editors, etc.
- When a process is run in the foreground, no other process can be run on the same terminal until the process is finished or killed.

Background Processes

Background Processes

- These are not connected to a terminal session and don’t expect any user input.

- They usually start automatically as part of the system functions/services.

- Examples are update managers, network managers, etc.

Commands

Foreground and background processes

```
[command] &
```

- user programs can be manually run as background processes.

- To run a process in the background, add an ampersand & at the end of the command.

- This launches the command in the background and returns control of the terminal to the user.

Foreground and background processes

fg %JID

- To bring a background or stopped (suspended) process to the foreground, use the fg command followed by the job ID.

bg %JID

- To resume a suspended process in the background, use the bg command

Daemon Process

Daemon Process

- A daemon is a type of background process that runs continuously, typically performing system-related tasks.
- Daemon processes are often started during system boot-up and run in the background without any user interaction.
- Daemons typically have no controlling terminal, which is indicated by a ? in the TTY field of the ps command's output.

Process Relationships

Process Relationships

Process Relationships

1. Parent Process
2. Child Process
3. Orphan Process
4. Zombie Process

Process Relationships

1 Parent Process

★ A parent process creates another process, either directly or indirectly.

★ Every process has a parent process, except for systemd.

★ When a process is created, it inherits various attributes from its parent process, such as its working directory.

Process Relationships

2 Child Process

- ★ A child process is one created by another process (its parent process).
- ★ Child processes inherit most of their attributes from their parent process but can also have their own unique attributes.
- ★ For example, a shell may create a child process that has its own environment variables and command-line arguments.

Process Relationships

3 Orphan Process

- ★ An orphan process is a process whose parent has terminated before them.
- ★ Orphan processes are re-parented to the init system, typically systemd, which continues to manage them.

Process Relationships

4 Zombie Process

★ A zombie process is a process that has completed execution but still has an entry in the process table, as its parent process has not yet retrieved its exit status.

★ Although zombies do not consume system resources like memory or CPU, they do occupy a slot in the process table

★ Zombie processes are usually terminated once the parent process retrieves the exit status.

Process Management

Process Management

Kill [signal] PID

- The kill command doesn't exactly “kill” processes;
- rather, it sends them signals.
- Signals are one of several ways that the operating system communicates with programs.

Process Management

Signals can be specified in three ways:

kill -15 PID	1_ By number
kill -TERM PID	2_ By name
kill -SIGTERM PID	3_ By name prefixed with SIG

Process Management

There are many signals, but the most common ones are:

Number	Signal	Description
9	KILL	Terminate immediately, hard kill.
15	TERM	Terminate whenever, soft kill.
19	STOP	Pause the process. (CTRL+Z)
18	CONT	Resume the process.

The default signal is 15 or TERM (terminate).

Process Management

killall [name]

- This command sends a signal to all instances (processes) of a specific program name.

- Unlike kill, which targets a specific PID, killall targets all processes with the specified name.

Let’s Play!

Who am I?

I am created when my parent dies unexpectedly, and now I’m reparented to PID 1. Who am I?

I am the very first software that runs when your system is powered on.

I wake up the CPU and hand over control to the loader.

Who am I?

I was terminated, but my entry remains in the process table because my parent didn’t clean me up. I linger as a ghost. Who am I?

I start running in the terminal and block your shell until I finish.

I demand your full attention. Who am I?

Hands on #2

1. Open a terminal and run `xlogo`.
2. Suspend (stop) the `xlogo` process.
3. Run another `xlogo` process in the background.
4. Search for the two Xlogo processes.
5. List the current shell jobs.
6. Bring the suspended job back to the foreground.
7. Open a new terminal and kill every `xlogo` process.

# IF YOU DON'T HAVE `xlogo` USE `sleep 10000`

Solution:

1. Open a terminal and run `xlogo`.

```
xlogo
```

2. Suspend (stop) the `xlogo` process.

```
cntrl+Z , kill -STOP PID
```

3. Run another `xlogo` process in the background.

```
xlogo &
```

4. Search for the two Xlogo processes.

```
ps -C xlogo
```

5. List the current shell jobs.

```
jobs
```

6. Bring the suspended job back to the foreground.

```
Fg %JID
```

7. Open a new terminal and kill every `xlogo` process.

```
killall xlogo
```

Thanks!