SHREE UTTAR GUJARAT BCA COLLEGE

Unit 2: Basic Linux Commands

2.1 Directory Navigation Commands (pwd, cd, mkdir, rmdir, ls, tree)

2.2 File Management Commands (cat, rm, cp, mv, touch)

2.3 File Permissions and Ownership (chmod, chgrp, chown, umask)

2.4 Common System Commands (who, whoami, man, echo, date, clear)

2.5 Text Processing Commands (head, tail, cut, sort, cmp, tr, uniq, wc, tee)

2.6 Introduction to Process

2.7 Process Control commands:ps, fg, bg, kill, sleep

2.8 Job Scheduling commands: at, batch, crontab

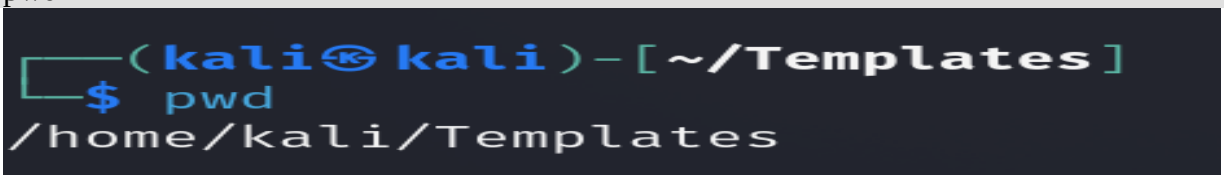
2.1 Directory Navigation Commands (pwd, cd, mkdir, rmdir, ls, tree)

Linux Commands	Functions
pwd	Shows the current location.
ls	List files and folders.
cd	Change working directory.
mkdir	Used to create new folder.
rmdir	Remove an empty folder.
tree	seeing the bigger picture in Linux

1. pwd (print working directory)

The **pwd** command shows the current location in the system. It tells you which folder you're currently in.

pwd



2. ls (list files and directories)

The **ls** command is used to list the files and directories in the current directory. It provides an overview of what is inside a folder.

ls

3. cd (change directory)

The **cd** command is used to move between folders. You can tell it exactly which folder you want to go to (like giving it an address), or you can use shortcuts to get around. Let's look into both the methods.

Moving around nearby folder

If you want to move into a folder that's within the one you're already in, you can just use its name. For instance, if you're in your **home** directory and want to reach **downloads.**

cd [directory name]

cd Downloads

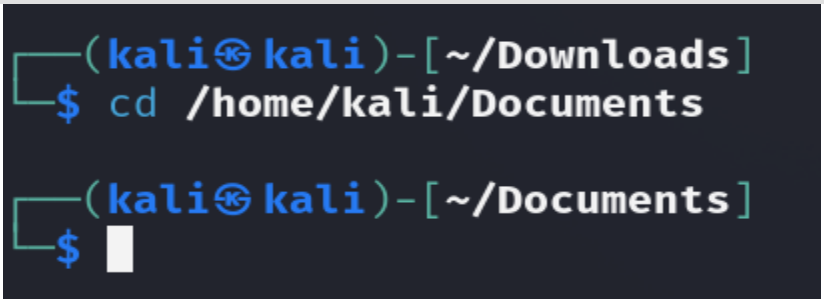
A. checking the current directory

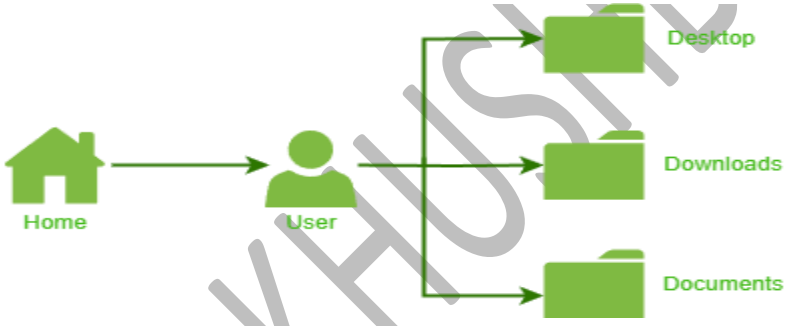
B. using cd command to change the directory
C. observe the updated directory

Going to a Specific Folder

Imagine telling someone the full address to find your house. Similarly, you can do the same by giving the complete path to the folder. For example, you want to access the **documents** folder inside the **username** folder.

```
cd [directory path]
cd  /home/username/documents
```

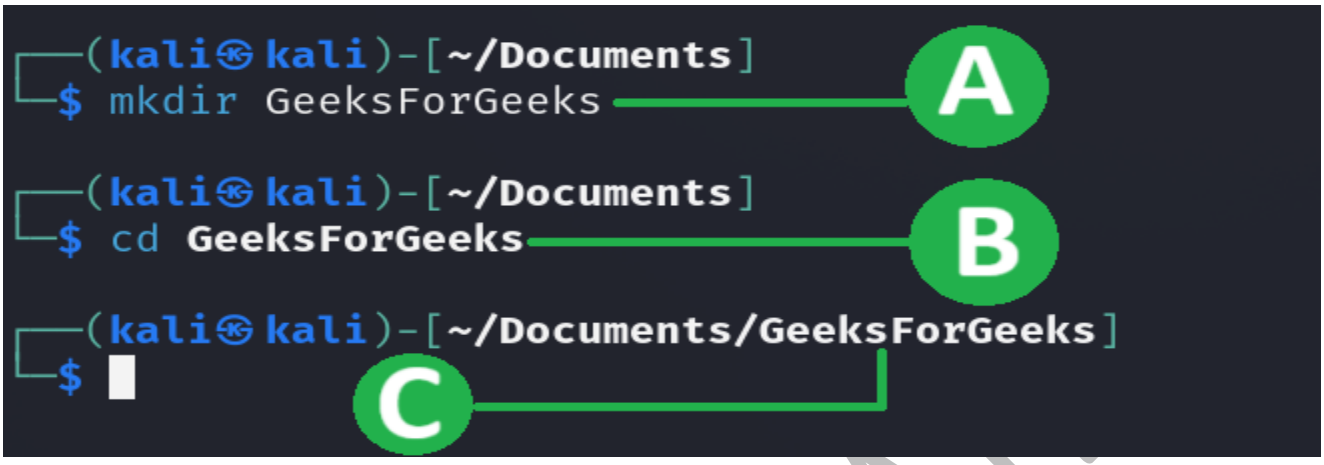




4. mkdir (make directory)

The mkdir command, an abbreviation for "make directory," allows you to create new folders within your existing file system. This provides a structured way to categorize and store your files.

```
mkdir [directory name]

mkdirGeeksForGeeks
```



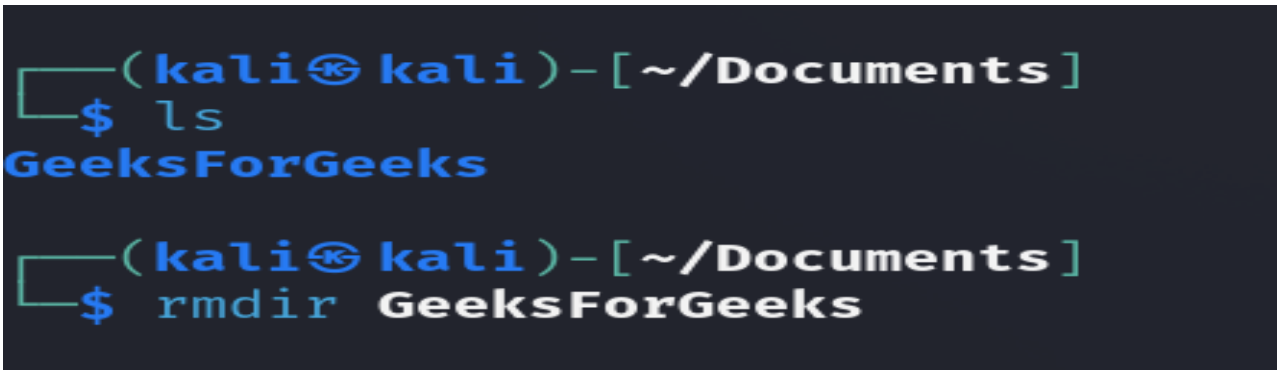
*A. User tried to create a new folder named **GeeksForGeeks***
B. changed the directory to newly created directory
C. The active directory is updated

5. rmdir (remove empty directory)

The rmdir command, short for "remove directory," enables you to delete empty directories. This is useful for cleaning up unused folders and maintaining a streamlined file system.
The syntax for rmdir is:

```
rmdir [directory_name]
```

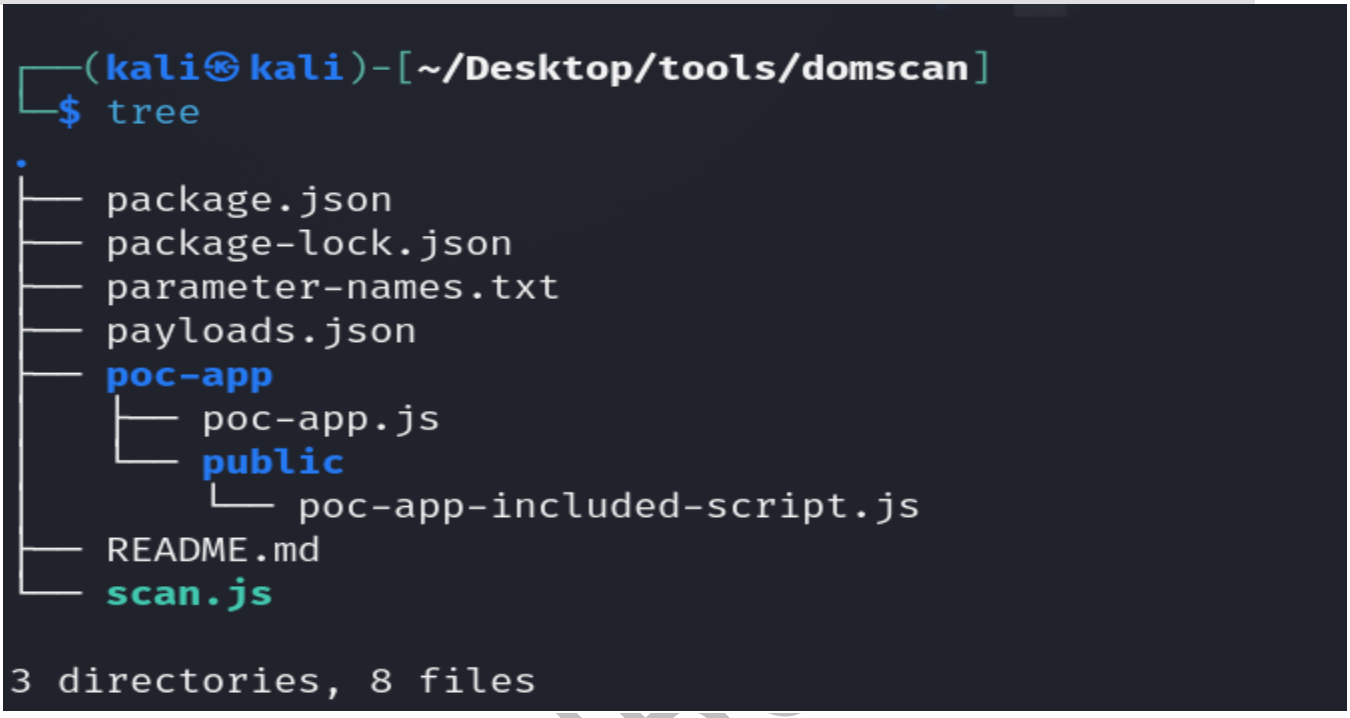
Note: rmdir can only delete empty directories.



6. Tree - seeing the bigger picture in Linux

This command isn't exactly for moving around, but it helps you see all the folders at once. It shows how all the folders are connected, like a family tree, so you can understand the bigger picture.

```
tree
```



All the files and folders of the active directory is listed.

2.2 File Management Commands (cat, rm, cp, mv, touch)

1. cat

1. How to View the Content of a Single File in Linux

The most basic use of 'cat' is to display the contents of a file on the terminal. This can be achieved by simply providing the filename as an argument:

Syntax:

```
catfile_name
```

Example: If our file_name = jayesh.txt

```
cat jayesh.txt
```

```
[jayeshkumar@localhost ~]$ ls
jayesh.txt
[jayeshkumar@localhost ~]$ cat jayesh.txt
hello GeeksforGeeks

[jayeshkumar@localhost ~]$
```

2. How to View the Content of Multiple Files in Linux

Syntax:

```
cat file_name1 file_name2
```

Example: If we have two files , file1 and file2.

```
cat file1 file2
```

```
[jayeshkumar@localhost ~]$ cat file1 file2
hello
hello this is file2
[jayeshkumar@localhost ~]$
```

3. How to View Contents of a File preceding with Line Numbers in Linux

Adding the -n option to cat introduces line numbers, making it convenient to identify and reference specific lines within the file.

Syntax:

```
cat -n file_name
```

Example: If our file_name is file2.

```
cat -n file2
```

```
[jayeshkumar@localhost ~]$ cat -n file2
     1  hello this is file2
     2  this is to display use of option `-n`
[jayeshkumar@localhost ~]$
```

4. How to Create a file and add content in Linux Using `cat` Command

If you want to create a new file or overwrite an existing file with new content, you can use 'cat' with the output redirection (`>`):

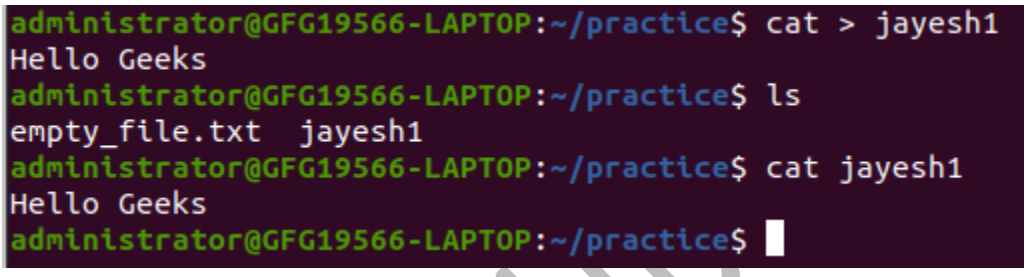
Syntax:

```
cat>newfile_name
```

Example: If we want to create a newfile_name = jayesh1

```
cat> jayesh1
ls
```

This will allow you to type text directly into the terminal, and when you press Ctrl + D, the entered text will be saved to new_file.txt.
`ls` command is used to display all files and directories in the current location.



5. How to Copy the Contents of One File to Another File in Linux

As the name suggests, 'cat' can concatenate multiple files into a single file.This example illustrates how to copy the entire content of "file1" into "file2" using the cat command along with redirection (>).

Syntax:

```
cat file1.txt file2.txt > merged_file.txt
```

This command combines the content of file1.txt and file2.txt into a new file named merged_file.txt.

6. Cat command can suppress repeated empty lines in output

The -s option comes in handy when dealing with files containing repeated empty lines. It suppresses these repetitions, providing a cleaner output.

Syntax:

```
cat -s file_name
```

Output

```
Will suppress repeated empty lines in output
```

7. How to Append the Contents of One File to the End of Another File

If you want to add the content of one file to another, 'cat' can be used along with the append (>>) operator:

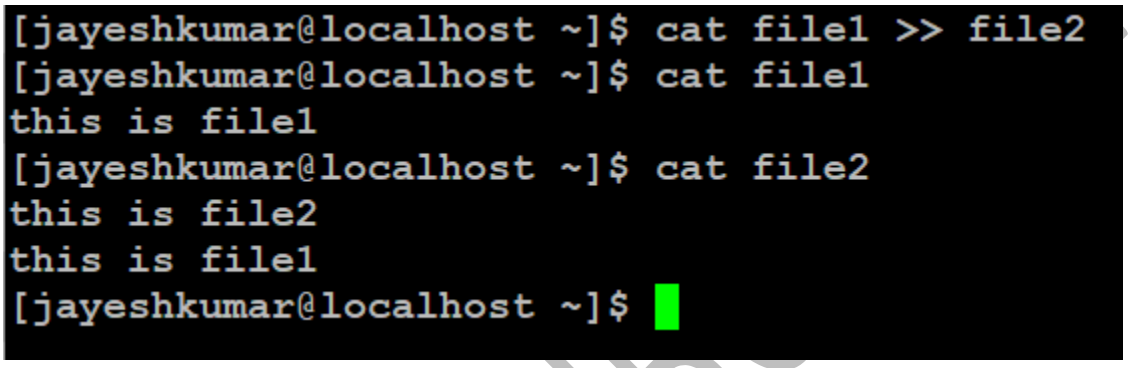
Syntax:

```
cat file_name1 >> file_name2
```

Example:

```
cat file1 >> file2
```

This will append the content of `file1` to the end of `file2`



8. How to Display Content in Reverse Order Using `tac` Command in Linux

The 'tac' command is the reverse of 'cat' and is used to display the content of a file in reverse order. The syntax is simple:

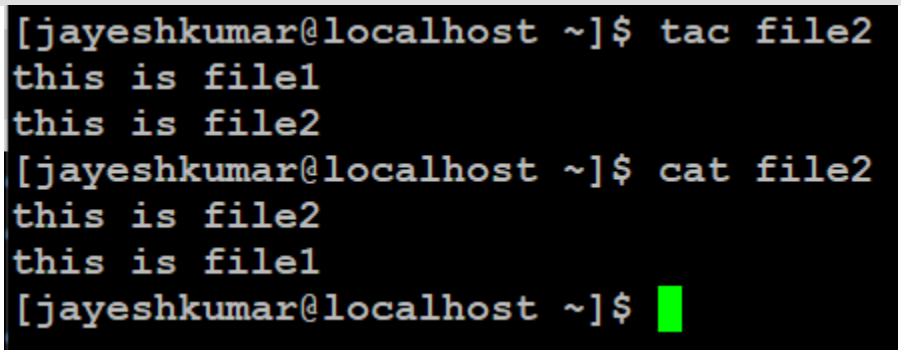
Syntax:

```
tacfile_name
```

Example:

This command will print the content of 'file2' in reverse order, displaying the last line first, followed by the second-to-last line, and so on.

```
tac file2
```



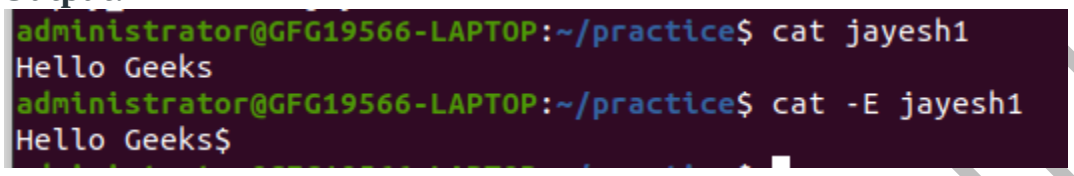
9. How to Highlight the End of Line in Linux

The '-E' option in the 'cat' command is used to highlight the end of each line.

Syntax:

```
cat -E "filename"
```

Output:



Displaying $ in the end of line

This will display the content of 'jayesh1' with a '$' character at the end of each line, indicating the line's terminate.

10. `-A` Command Line Option in `cat` Command in Linux

The '-A' option allows you to combine the effects of '-v', '-E', and '-T' options. Instead of writing '-vET' in the command, you can use '-A':

Syntax:

```
cat -A  "filename"
```

Explanation:

- The -v option creates non-printing characters visible (except for tabs and line breaks).
- The -E option emphasizes the end of each line with a $.
- The -T option means that if the file contains tab characters, they will be displayed as ^I

This will display the content of 'filename' with non-printing characters visible, line endings highlighted, and tabs displayed as '^I'.

11. How to Open Dashed Files in Linux Using `cat` Command

To open a file with a dash at the beginning of its name, use the '--' option:

Syntax:

```
cat -- "-dashfile"
```

Example:

```
cat -- "-jayesh2"
```

```
administrator@GFG19566-LAPTOP:~/practice$ cat -jayesh2
cat: invalid option -- 'j'
Try 'cat --help' for more information.
administrator@GFG19566-LAPTOP:~/practice$ cat -- "-jayesh2"
Hello GeeksforGeeks
```

Displaying content inside a file starting with `-`
This will display the content of a file named "-jayesh2"

12. Cat command if the file has a lot of content and can't fit in the terminal.

Syntax:

```
cat "filename" | more
```

Output:

Will show that much content, which could fit in terminal and will ask to show more.

13. Merge Contents of Multiple Files Using `cat` Command

To merge the contents of multiple files into a single file, use the redirection ('>')

Syntax:

```
cat "filename1" "filename2" "filename3" > "merged_filename"
```

Example:

```
cat "file1" "file2" "file3" > "merged123"
```

This will concatenate the contents of "file1" "file2" "file3" into "merged123".
merging content of multiple files into single file

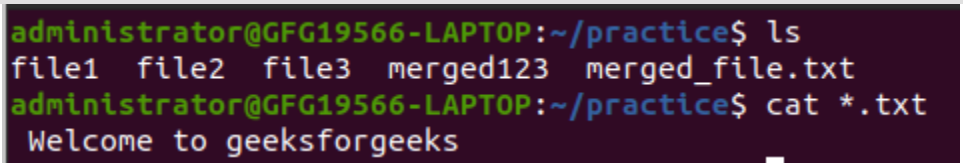
```
administrator@GFG19566-LAPTOP:~/practice$ ls
file1  file2  file3  merged123  merged_file.txt
administrator@GFG19566-LAPTOP:~/practice$ cat file1 file2 file3 merged123
this is file 1
this is file 2

this is file 3

this is exple to merge files.
administrator@GFG19566-LAPTOP:~/practice$ cat "file1" "file2" "file3" > "merged123"
administrator@GFG19566-LAPTOP:~/practice$ cat merged123
this is file 1
this is file 2

this is file 3
```

14. Display Content of All Text Files in a Folder Using `Cat` Command

To display the content of all text files in a folder, use the wildcard ('*.txt'):

Syntax:

```
cat *.txt
```



Displaying all file with extension ".txt"
Will show the content of all text files present in the folder.

15. Cat Command to Append to an Existing File:

To append text to an existing file, use the '>>' operator along with 'cat':

Syntax:

```
cat>> geeks.txt
The newly added text.
```

This will append the text "The newly added text." to the end of the 'geeks.txt' file.

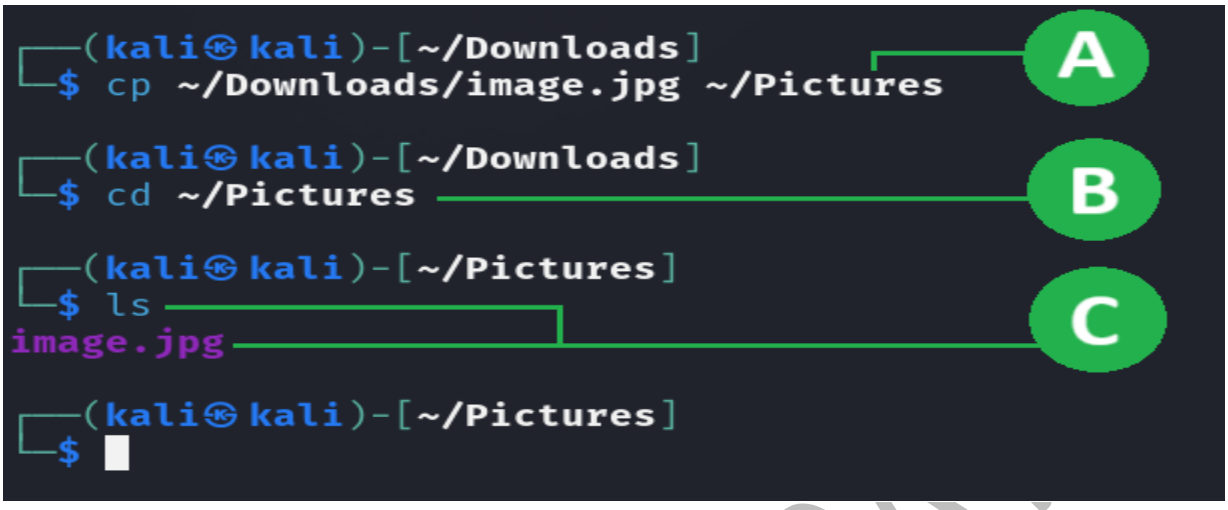
2. cp (copy)

The **cp** command acts like a duplicator, creating a copy of a file in a new location.

```
cp [source_file] [location]
```

For instance, to copy "image.jpg" from Downloads to Pictures while keeping the original, you'd use

```
cp ~/Downloads/image.jpg ~/Pictures
```



A. Executing the copying command.
B. Changing the current active directory for checking.
C. Using the list command, we can observe that the image.jpg was copied.

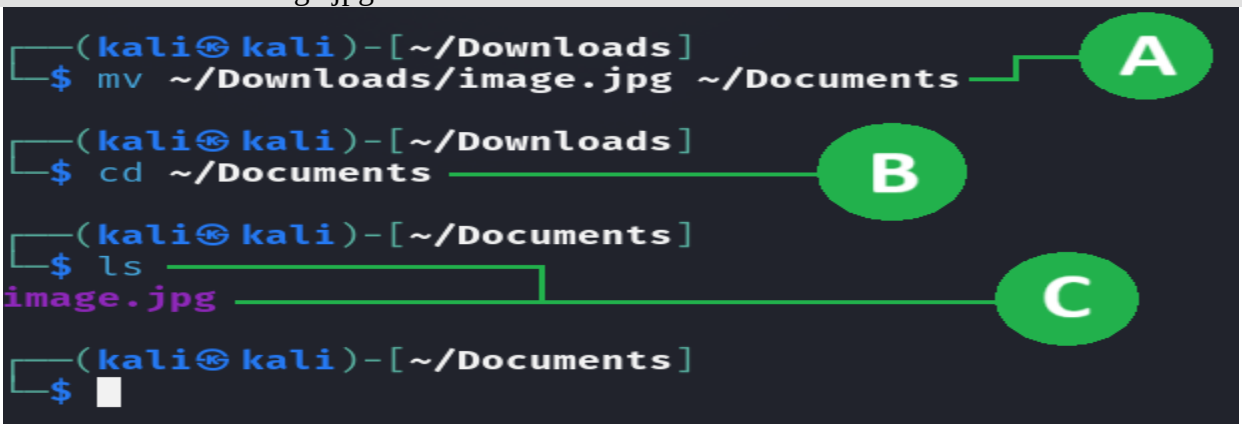
3. mv (move)

The **mv** command is like a handy mover, allowing you to relocate files from one folder to another.

```
mv [source_file] [location]
```

For instance, to move a file named "image.jpg" from your Downloads folder to Documents, you'd use

```
mv  ~/Downloads/image.jpg  ~/Pictures
```



A. Executing the mv (move) command.
B. Changing the current active directory.

C. After executing ls (list) command, we can observe that the file has been transferred.

4. touch

To update the access and modification times of a file, use touch filename:

Example

```
touch file1.txt
```

Options

The touch command has options to change how it works:

- -a - Update only when the file was last read
- -m - Update only when the file was last changed
- -t - Set the timestamp to a specific time
- -c - Do not create any files

-a Option: Change Access Time

The -a option lets you update the access time of a file.

Example: Change Access Time

```
touch -a file.txt
```

-m Option: Change Modification Time

The -m option lets you update only the modification time of a file.

Example: Change Modification Time

```
ls -l

-rw-r--r-- 1 user 197609 208 Apr  9 06:24 my_file.txt

touch -m my_file.txt
```

```
ls -l

-rw-r--r-- 1 user 197609 208 Apr  9 08:25 my_file.txt
```

-t Option: Set Specific Timestamp

The -t option allows you to set the timestamp to a specific time.

Example: Set Specific Timestamp

```
ls -l

-rw-r--r-- 1 user 197609 208 Apr  9 06:24 my_file.txt

touch -t 202501010000 my_file.txt

ls -l

-rw-r--r-- 1 user 197609 208 Jan  1 00:00 my_file.txt
```

-c Option: Do Not Create Files

The -c option tells touch not to create any files if they do not exist.

This is useful when you want to update timestamps without accidentally creating new files.

Example: Do Not Create Files

```
touch -c non_existent_file.txt
```

2.3 File Permissions and Ownership (chmod, chgrp, chown, umask)

1. chmod

What is chmod?

Chmod stands for **Change Mode**.It is a command used in **Linux/Unix** to **change the permissions** (read, write, execute) of a file or directory.

Linux File Permissions Overview

Each file and directory has **three types of users**:

User Type	Description
u	User (Owner of the file)
g	Group (Users in the same group)
o	Others (Everyone else)

And **three types of permissions**:

Symbol	Permission	Description
r	Read	View the file contents
w	Write	Modify the file
x	Execute	Run the file (if it's a script/program)

Syntax of chmod

chmod [options] mode file_name

There are **two ways** to use chmod:

1. Symbolic Mode

chmodu+x filename # Add execute permission to user
chmodg-w filename # Remove write permission from group
chmodo=r filename # Set others' permission to read only

Operator	Meaning

Operator	Meaning
+	Add permission
-	Remove permission
=	Set exact permission

2. Numeric Mode (Octal Mode)

Each permission has a value:

Permission	Value
r	4
w	2
x	1

Calculate total for each user type:

Example:

chmod 755 filename

- 7 = 4+2+1 (rwx) for **user**
- 5 = 4+0+1 (r-x) for **group**
- 5 = 4+0+1 (r-x) for **others**

Common Examples

Command	Description
chmod 777 myfile	Full permission to everyone (rwxrwxrwx)
chmod 644 myfile	User: read/write, Group/Others: read-only
chmod +x script.sh	Make a script executable

Command	Description
chmodugo-rwx file	Remove all permissions from everyone

View File Permissions

Use ls -l to see permissions:

ls -l filename

Example output:

-rwxr--r-- 1 user group1234 Jul 2310:00 filename

Breakdown:

- -rwxr--r--
  - rwx → User
  - r-- → Group
  - r-- → Others

2. chgrp

What is chgrp?

chgrp stands for **Change Group**.
It is used to **change the group ownership** of a file or directory.

Every file in Linux has:

- **Owner** (user who created it)
- **Group** (a set of users who may share access)
- **Permissions** for user, group, and others

Syntax of chgrp

chgrp [options] group_name filename

- group_name: Name of the group you want to assign
- filename: Name of the file or directory

Example Usage

chgrp students report.txt

This command sets the group **students** as the new group owner of **report.txt**.

How to View Group Ownership?

Use:

ls -l filename

Example output:

-rw-r--r--1 user students 1200 Jul 2310:00 report.txt

- Here, **students** is the group that owns the file.

Change Group for a Directory

chgrp developers myproject/

To change group for all files and subdirectories **recursively**:

chgrp -R teamnamefoldername/

Common Options

Option	Description
-R	Recursively change group of directories
--help	Show help for command
--verbose	Show details of what is being changed

3. chown

What is chown?

- chown stands for **change ownership**.
- It is used to **change the owner or group** of a file or directory.

Syntax:

chown [OPTIONS] NEW_OWNER[:NEW_GROUP] FILE

Common Options:

Option	Description
-R	Change ownership recursively
-v	Verbose – show what's being changed
--help	Show help message

Example Explanation:

- chown user1 file.txt – Changes owner to user1.
- chown user1:group1 file.txt – Changes both owner and group.

4. umask

What is umask?

- umask stands for **User File-Creation Mode Mask**.
- It defines **default permission bits** for new files and directories.
- It **subtracts permissions** from the system default when a new file or folder is created.

Default Permissions in Linux:

File Type	Default Permission
Files	666 (rw-rw-rw-)

File Type Default Permission

Directories 777 (rwxrwxrwx)

Note: Executable (x) is not set by default for files.

Syntax:

umask [mask]

- Without any arguments: displays current mask.
- With a numeric value: sets a new mask.

How umask Works:

Actual Permissions = Default Permissions - umask

Example:

- If umask is 022:
  - Files: 666 - 022 = **644 (rw-r--r--)**
  - Dirs: 777 - 022 = **755 (rwxr-xr-x)**

Common umask Values:

umask	File Permission	Directory Permission
022	644 (rw-r--r--)	755 (rwxr-xr-x)
027	640 (rw-r-----)	750 (rwxr-x---)
077	600 (rw-------)	700 (rwx------)

1: View Current umask

umask

2: Create File with Default umask

```
touch file1.txt
ls -l file1.txt
```

3: Change umask temporarily

```
umask 077
touch file2.txt
mkdir dir1
ls -l file2.txt
```


2.4 Common System Commands (who, whoami, man, echo, date, clear)

1. Who – See who is logged in

✅ **Purpose:**

Displays a list of users currently logged into the system.

 Example:

Who

Output Example:

student1 tty1 2025-07-25 08:15

2. whoami – Display current user

✅ **Purpose:**

Shows the **username** of the current user.

 Example:

whoami

Output Example:

student1

3. man – Manual pages/help

✅ **Purpose:**

Displays the **manual/help** page for any Linux command.

 Example:

man ls

✅ Press q to exit the manual.

4. echo – Display a line of text

✅ **Purpose:**

Prints messages, variable values, or outputs to screen.

 Examples:

echo Hello World

Output:

Hello World
name="Linux"
echo $name

Output:

Linux

5. date – Show current date and time

✅ **Purpose:**

Displays the current system date and time.

 Example:

date

Output Example:

Thu Jul 25 08:30:00 IST 2025

date +"%d-%m-%Y"

Output:

25-07-2025

6. clear – Clear terminal screen

✅ **Purpose:**

Clears all previous output from the terminal.

 Example:

clear

✅ Terminal screen becomes blank, but previous commands can still be scrolled up.

2.5 Text Processing Commands (head, tail, cut, sort, cmp, tr, uniq, wc, tee)

1. head – View first few lines of a file

Purpose:

Displays the **first 10 lines** by default.

Syntax:

head [file]
head -n [number] [file]

Example:

head filename.txt
head -n 5 filename.txt

2. tail – View last few lines of a file

Purpose:

Displays the **last 10 lines** by default.

Syntax:

tail [file]
tail -n [number] [file]

Example:

tail filename.txt
tail -n 3 filename.txt

3. cut – Extract specific columns/fields

Purpose:

Cuts out sections from each line of a file.

Syntax:

```
cut -c [position] [file]      # by character
cut -d [delimiter] -f [field] [file]  # by field
```

Examples:

```
cut -c 1-5 filename.txt
cut -d "," -f2 student.csv
```

4. sort – Sort lines of text files

Purpose:

Sorts lines alphabetically or numerically.

Syntax:

```
sort [file]
sort -n [file]       # numeric sort
sort -r [file]       # reverse order
```

Example:

```
sort names.txt
sort -n marks.txt
```

5. cmp – Compare two files byte by byte

Purpose:

Tells whether files are same or different.

Syntax:

cmp file1 file2

Example:

cmp file1.txt file2.txt

Output: First difference position or nothing (if identical).

6. tr – Translate or delete characters

Purpose:

Used for replacing or removing characters.

Syntax:

tr [set1] [set2]

Example:

echo "HELLO" | tr 'A-Z' 'a-z'

Output: hello

7. uniq – Remove duplicate lines

Purpose:

Filters out repeated lines (needs sorted input).

Syntax:

uniq [file]
uniq -c [file] # show counts

Example:

sort data.txt | uniq

8. wc – Word, line, character count

Purpose:

Counts lines, words, and characters.

Syntax:

wc [file]
wc -l [file] # lines
wc -w [file] # words
wc -c [file] # bytes

Example:

wc notes.txt

9. tee – Read and write to file and output

Purpose:

Reads from input and writes to both **terminal and file**.

Syntax:

command | tee file

Example:

echo "Backup Complete" | tee log.txt

Output: Displays message and saves it to log.txt.

2.6 Introduction to Process

What Is a Process?

A process in Linux is a running instance of a program.When you execute a command or launch an application, the system creates a process to handle that task.Each process has its own memory space, system resources, and execution context.Processes can be short-lived (like listing files) or long-running (like a web server).

Types of Processes

Linux processes can be categorized into:

- **Foreground Processes**: These run interactively and occupy the terminal until completion.
- **Background Processes**: These run independently of the terminal, allowing users to continue other tasks.

Process States

- **Running**: Actively executing or ready to execute.
- **Sleeping**: Waiting for an event or resource.
- **Stopped**: Execution has been halted.
- **Zombie**: Process has completed execution but still has an entry in the process table.

Process Management Commands

- ps: Displays current processes.
- top: Real-time view of running processes and resource usage.

- kill: Sends signals to processes, commonly used to terminate them.
- nice/renice: Adjusts the priority of a process.
- bg/fg: Moves processes between background and foreground.
- 2.7 Process Control commands:ps, fg, bg, kill, sleep

2.7 Process Control commands: ps, fg, bg, kill, sleep

1. ps – Display current processes

Purpose:

Shows **currently running processes** for the user or system.

Syntax:

ps# Show your processes
ps -e # Show all processes
ps -ef# Full-format list (detailed)

Example:

ps
ps -ef | grep firefox

✓ Use to **find process IDs (PID)** and check running programs.

2. sleep – Delay execution

Purpose:

Pauses the terminal for a given number of **seconds**.

Syntax:

sleep [time]

Example:

```
sleep 5
echo"Finished waiting 5 seconds"
```

Used in scripting to create **delays** or **simulate wait times**.

□ 3. fg – Resume a background job in foreground

Purpose:

Brings a **suspended or background** job to the foreground.

Syntax:

```
fg [%job_id]
```

Example:

```
gedit&
fg
```

Use jobs to list background jobs first.

4. bg – Resume a job in background

Purpose:

Resumes a **stopped job** and continues it in the **background**.

Syntax:

```
bg [%job_id]
```

Example:

```
sleep 60
# Press Ctrl+Z to pause
```

bg

The job continues running, and you can keep using the terminal.

5. kill – Terminate a process

Purpose:

Stops a running process by **PID**.

Syntax:

kill [PID]
kill -9 [PID] # Force kill

Example:

ps
kill 1234

Replace 1234 with the actual process ID.

2.8 Job Scheduling commands: at, batch, crontab

1. at – Schedule a one-time task

Purpose:

Schedules a **command or script to run once at a specific time**.

Syntax:

at [TIME]

Requirements:

The atd (at daemon) service must be running.

Example:

at 14:30

Then type:

echo"Reminder: Attend class">> notes.txt
<Ctrl+D>

This command will run at 2:30 PM and append the message to notes.txt.

View scheduled jobs:

atq

Remove a job:

atrm [job_number]

2. batch – Run a task when system load is low

Purpose:

Schedules a **one-time job** to run **when the system load is low**.

Syntax:

batch

Example:

batch

Then enter:

echo"Backup complete">> backup_log.txt
<Ctrl+D>

Job will run automatically when the system is idle.

□ 3. crontab – Schedule recurring tasks

Purpose:

Schedules **repeating jobs** (daily, weekly, monthly, etc.).

Syntax:

```
crontab -e   # Edit cron jobs
crontab -l   # List user cron jobs
crontab -r   # Remove all cron jobs
```

Crontab Time Format:

```
sql
CopyEdit
*****  command
```