

Kathmandu University

Department of Computer Science and Engineering



Dhulikhel, Kavrepalanchowk

Lab Report: 01
COMP 307

Submitted by:
Aayusha Shrestha
Roll No: 50
Level: UNG CS (III/I)

Submitted to:
Rabina Shrestha

Department of Computer Science and Engineering

Submission Date:
12/10/2025

1. Introduction

1.1 What is Linux?

Linux is an open-source operating system based on UNIX. It is known for its stability, security, and flexibility, making it widely used in servers, networking, cloud computing, and software development. Being open-source, Linux allows users to view, modify, and distribute the source code freely.

1.2 The Linux Hierarchical File System

Linux uses a tree-like directory structure. The top-most directory, called the root directory (/), contains essential system directories such as /home, /etc, /bin, and /usr. Users typically work within their own directories under /home/username, which ensures organized file management and separation between system files and user data.

1.3 Importance of Linux Commands

Linux commands provide a powerful interface for controlling and managing the system efficiently. They allow users to navigate directories, manage files and folders, monitor processes, and perform system administration tasks. Mastery of these commands is essential for working effectively in a Linux environment, whether for development, administration, or troubleshooting.

2. Linux Commands

2.1 Command: pwd

The *pwd* command in Linux stands for “print working directory”. Its primary function is to display the absolute path of the current directory within which we are currently operating. This helps users understand their location within the file system hierarchy..
For example: /home/aayusha/Documents

It's useful when navigating through multiple folders.

Output:

/home/aayusha

```
c:\ aayusha@DESKTOP-9N9URUN ~$ pwd  
/home/aayusha  
aayusha@DESKTOP-9N9URUN ~$ |
```

2.2 Command: ls

This command is used to list all the files and directories in the current working directory. In this case, when the command was executed in the root directory (/root), nothing was displayed because the directory is currently empty. This simply means there are no visible files or folders present in that directory. The command only shows non-hidden files, so any files starting with '.' will not appear. This helps the user quickly check which files and folders are available without showing system or hidden configuration files. Thus, an empty output from *ls* is normal in a directory with no files.

Output:

```
# No files or directories in /root, so nothing is shown
```

```
aayusha@DESKTOP-9N9URUN ~$ ls  
aayusha@DESKTOP-9N9URUN ~$ |
```

2.3 Command: ls -a

The `ls -a` command lists all files in the directory, including hidden files that start with a dot (.). Even if a directory appears empty with `ls`, `ls -a` will show hidden files like `.bashrc` or `.profile`. In this example, running `ls -a` in /root showed the special entries `.` (current directory) and `..` (parent directory), along with any hidden configuration files. Hidden files are usually system or environment configuration files. This command is useful to check for any hidden files that may not be visible with a simple `ls`. If there are no hidden files other than `.` and `..`, the directory is truly empty.

Output:

... .bashrc Documents file.txt

```
aayusha@DESKTOP-9N9URUN:~$ ls -a
. . . . bash_logout .bashrc .cache .landscape .motd_shown .profile .sudo_as_admin_successful
aayusha@DESKTOP-9N9URUN:~$ |
```

2.4 Command: ls -l

The `ls -l` command displays files and directories in long listing format with detailed information, including:

- Permissions (read/write/execute)
- Owner and group
- File size
- Last modified date and time
- Name of the file or directory

Since the /root directory is empty, there are no files or folders to display. Even though the output is empty, running this command is useful in other directories to check the properties and metadata of files. It is a more informative way to view files compared to a simple `ls`, especially for managing permissions and ownership.

Output:

Total:0

```
aayusha@DESKTOP-9N9URUN:~$ ls -l
total 0
aayusha@DESKTOP-9N9URUN:~$ |
```

2.5 Command: cd

The *cd* command allows you to move between directories.

- *cd ..* → moves one level up in the directory tree.
- *cd /path/to/folder* → moves directly to a specific folder.
- *cd* without any arguments → takes you to your home directory.

It helps in navigating the file system efficiently. Here, represents the home directory of the current user (in this case, root). This is a quick way to return to the home directory from anywhere in the file system. The command does not produce any output; it only changes the current directory.

```
aayusha@DESKTOP-9N9URUN:~$ cd  
aayusha@DESKTOP-9N9URUN:~$ |
```

2.6 Command: mkdir

The *mkdir* command is used to create a new directory in the current working directory.

Example: *mkdir Projects* creates a folder named *Projects* immediately in the current directory. To create nested directories in one command, you can use: *mkdir -p folder1/folder2*. The command does not show any output when successful, it silently creates the directory. To verify that the directory was created, you can run *ls* or *ls -l* in the same location.

ls will list the directory name: *Projects*

```
aayusha@DESKTOP-9N9URUN:~$ mkdir Projects  
aayusha@DESKTOP-9N9URUN:~$ ls  
Projects  
aayusha@DESKTOP-9N9URUN:~$ |
```

2.7 Command: rmdir

The *rmdir* command deletes an empty directory. It cannot delete directories that contain files; attempting to do so will produce an error. Only directories with no contents can be

removed using this command. It is useful for cleaning up unnecessary empty folders. Example: *rmdir Projects* deletes the Projects folder if it is empty. If successful, there is no output, if the directory contains files, an error is displayed.

```
aayusha@DESKTOP-9N9URUN:~$ ls  
Projects  
aayusha@DESKTOP-9N9URUN:~$ rmdir Projects  
aayusha@DESKTOP-9N9URUN:~$ ls  
aayusha@DESKTOP-9N9URUN:~$ |
```

2.8 Command: rm

The *rm* command is used to delete files permanently.

Example: *rm testfile.txt* deletes the file named *testfile.txt*. It does not move files to a trash/recycle bin, so deleted files cannot be recovered easily. It is useful for removing unwanted files quickly. For safety, options like *-i* can be used to confirm deletion interactively. If successful, there is no output, an error is shown if the file does not exist.

```
aayusha@DESKTOP-9N9URUN:~$ ls  
testfile.txt  
aayusha@DESKTOP-9N9URUN:~$ rm testfile.txt  
aayusha@DESKTOP-9N9URUN:~$ ls  
aayusha@DESKTOP-9N9URUN:~$ |
```

2.9 Command: rm -r

The *rm -r* command deletes directories and all their contents recursively. It can remove folders even if they contain files and subdirectories.

Example: *rm -r Aayusha* deletes folder1 along with everything inside it. This command is dangerous if used carelessly because deletion is permanent. It is useful for cleaning up entire project directories. No output is shown if deletion is successful, errors appear if the directory does not exist.

```
aayusha@DESKTOP-9N9URUN:~$ ls
Aayusha  Projects
aayusha@DESKTOP-9N9URUN:~$ rm -r Aayusha
aayusha@DESKTOP-9N9URUN:~$ ls
Projects
aayusha@DESKTOP-9N9URUN:~$ |
```

2.10 Command: touch

The *touch* command creates a new empty file or updates the timestamp of an existing file.

Example: *touch new.txt* creates a file named *new.txt*. It is commonly used to create placeholder files quickly. If the file already exists, the command updates the last modified time without changing the content. This command produces no output, but listing the directory with *ls* will show the new file. Useful in scripting or preparing file structures for projects.

```
aayusha@DESKTOP-9N9URUN:~$ touch new.txt
aayusha@DESKTOP-9N9URUN:~$ ls
Projects  new.txt
aayusha@DESKTOP-9N9URUN:~$ |
```

2.11 Command: cat

The *cat* command displays the contents of a file in the terminal.

Example: *cat new.txt* prints everything in *new.txt* to the screen. It is best for viewing small files quickly. For long files, use less or more to scroll page by page. It does not modify the file, it only shows the content. Output is the actual content of the file.

```
aayusha@DESKTOP-9N9URUN:~$ echo "Hello Linux" > new.txt
aayusha@DESKTOP-9N9URUN:~$ cat new.txt
Hello Linux
aayusha@DESKTOP-9N9URUN:~$ |
```

2.12 Command: echo

The *echo* command prints text to the terminal or writes it to a file.

Example: *echo “Hello Linux”* prints Hello Linux on the screen. You can also redirect output to a file: *echo “Hello” > file.txt*. It is useful in scripts to display messages or create content for files. By default, it writes to standard output, using *>* or *>>* writes to files. Output is the exact text provided in the command.

```
aayusha@DESKTOP-9N9URUN:~$ echo "Hello Linux"
Hello Linux
aayusha@DESKTOP-9N9URUN:~$ |
```

2.13 Command: cp

The *cp* command copies files or directories from one location to another.

Example: `cp a.txt b.txt` creates a copy of `a.txt` named `b.txt`. It is useful for making backups or duplicating files. Options like `-r` can be used to copy directories recursively. If the destination file already exists, it will be overwritten. No output is shown if the copy is successful; errors occur if the source file does not exist.

```
aayusha@DESKTOP-9N9URUN:~$ cp new.txt Aayusha.txt
aayusha@DESKTOP-9N9URUN:~$ cat Aayusha.txt
Hello Linux
aayusha@DESKTOP-9N9URUN:~$ |
```

2.14 Command: nano/vi/jed

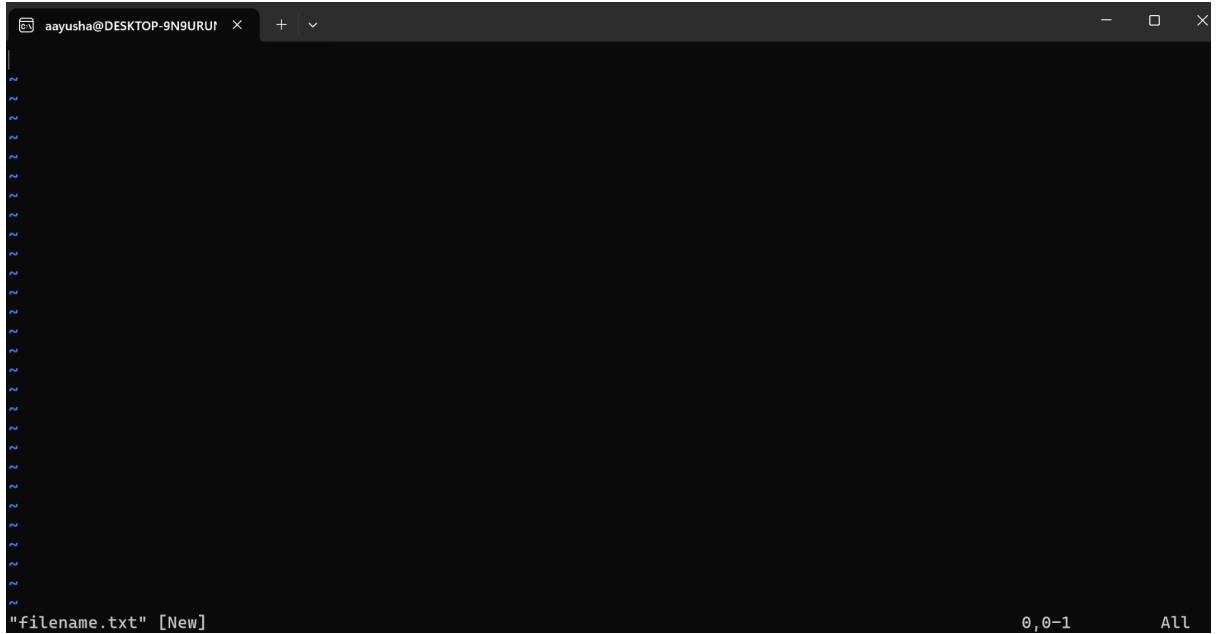
These are terminal-based text editors used to create and edit files directly from the command line.

- **nano:** Beginner-friendly, commands are displayed at the bottom of the editor.
Example: `sudo nano filename.txt`



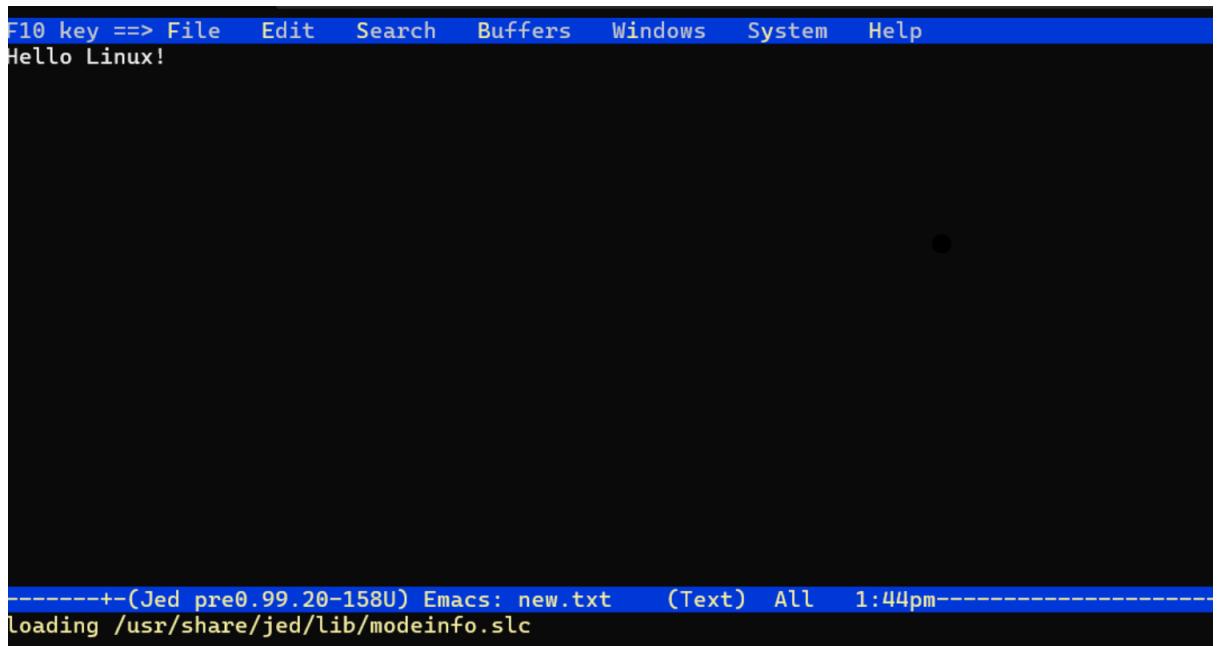
- **vi:** Powerful but a bit complex, requires switching between command and insert modes.

Example: vi filename.txt



A screenshot of a terminal window titled "aayusha@DESKTOP-9N9URUF ~". The window contains a single character, a tilde (~), repeated vertically down the page. At the bottom of the screen, the text "'filename.txt' [New]" is displayed, along with status information "0,0-1" and "All".

- **jed:** Lightweight editor, good for coding and quick edits.
Example: jed myfile.txt



A screenshot of the jed editor interface. The top menu bar includes "F10 key ==> File Edit Search Buffers Windows System Help". The main workspace displays the text "Hello Linux!". The bottom status bar shows the path "-----+-(Jed pre0.99.20-158U) Emacs: new.txt (Text) All 1:44pm-----" and the message "Loading /usr/share/jed/lib/modeinfo.slc".

2.15 Command: mv

The *mv* command moves or renames files and directories.

Example: *mv old.txt new.txt* renames *old.txt* to *new.txt*.

Example: *mv file.txt folder/* moves *file.txt* into the *folder* folder. It can be used for both moving files to a different location and renaming them. If the destination exists, it will be overwritten without warning unless *-i* is used. No output appears if successful, errors occur if the source does not exist.

```
aayusha@DESKTOP-9N9URUN:~$ ls
Aayusha.txt Projects new.txt
aayusha@DESKTOP-9N9URUN:~$ mv Aayusha.txt Projects/
aayusha@DESKTOP-9N9URUN:~$ ls
Projects new.txt
aayusha@DESKTOP-9N9URUN:~$ cd Projects/
aayusha@DESKTOP-9N9URUN:~/Projects$ ls
Aayusha.txt
aayusha@DESKTOP-9N9URUN:~/Projects$ |
```

2.16 Command: uname -a

The *uname -a* command displays system information. It shows kernel name, hostname, kernel version, OS type, architecture, and other details.

Example output: Linux DESKTOP-9N9URUN 6.6.87.2-microsoft-standard-WSL2 #1 SMP PREEMPT_DYNAMIC Thu Jun 5 18:30:46 UTC 2025 x86_64 x86_64 x86_64
GNU/Linux

Useful for checking system configuration and environment details. This command does not modify the system. Output appears directly on the terminal.

```
aayusha@DESKTOP-9N9URUN:~/Projects$ uname -a
Linux DESKTOP-9N9URUN 6.6.87.2-microsoft-standard-WSL2 #1 SMP PREEMPT_DYNAMIC Thu Jun 5 18:30:46 UTC 2025 x86_64 x86_64
x86_64 GNU/Linux
aayusha@DESKTOP-9N9URUN:~/Projects$ |
```

2.17 Command: top

The *top* command displays real-time system process information. It shows CPU, memory usage, and active processes dynamically.

Example: running top continuously updates the display until you press *q* to quit. Useful for monitoring system performance and identifying resource-heavy processes. The display includes PID, user, priority, CPU. It is a live interactive command, not a static output.

```
aayusha@DESKTOP-9N9URUN:~/Projects$ top
top - 16:59:55 up 12 min, 1 user,  load average: 0.00, 0.01, 0.00
Tasks: 24 total, 1 running, 23 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
Mib Mem : 7784.2 total, 7280.8 free, 531.0 used, 110.4 buff/cache
Mib Swap: 2048.0 total, 2048.0 free, 0.0 used. 7253.2 avail Mem

PID USER      PR  NI    VIRT   RES   SHR S %CPU %MEM TIME+ COMMAND
 568 aayusha  20   0  9272  5376 3328 R  0.3  0.1  0:00.01 top
  1 root     20   0 21732 12468 9268 S  0.0  0.2  0:00.42 systemd
  2 root     20   0  3060  1792 1792 S  0.0  0.0  0:00.00 init-systemd(Up)
  7 root     20   0  3076  1792 1792 S  0.0  0.0  0:00.00 init
 50 root     19  -1 50452 15616 14720 S  0.0  0.2  0:00.11 systemd-journal
 99 root     20   0 25360  6528 4864 S  0.0  0.1  0:00.07 systemd-udevd
109 systemd+ 20   0 21452 12032 10112 S  0.0  0.2  0:00.06 systemd-resolve
110 systemd+ 20   0  91020  7552 6784 S  0.0  0.1  0:00.05 systemd-timesyncd
179 turnserver+ 20   0 1988792 17536 12160 S  0.0  0.2  0:01.12 turnserver
180 root     20   0  4236  2432 2304 S  0.0  0.0  0:00.00 cron
181 message+ 20   0  9532  4992 4480 S  0.0  0.1  0:00.09 dbus-daemon
188 root     20   0 17964  8320 7424 S  0.0  0.1  0:00.07 systemd-logind
191 root     20   0 1755840 11520 10240 S  0.0  0.1  0:00.10 wsl-pro-service
196 root     20   0  3160  1920 1792 S  0.0  0.0  0:00.00 getty
210 syslog   20   0 222598  5248 4224 S  0.0  0.1  0:00.05 rsyslogd
212 root     20   0  3116  1792 1664 S  0.0  0.0  0:00.00 getty
221 root     20   0 107012 22528 13184 S  0.0  0.3  0:00.08 unattended-upgrader
328 root     20   0  3064  896  896 S  0.0  0.0  0:00.00 SessionLeader
329 root     20   0  3080  1024  896 S  0.0  0.0  0:00.07 Relay(330)
330 aayusha  20   0  6072  5248 3584 S  0.0  0.1  0:00.08 bash
331 root     20   0  6688  4480 3712 S  0.0  0.1  0:00.00 login
382 aayusha  20   0 20296 11264 9216 S  0.0  0.1  0:00.04 systemd
```

2.17 Command: df -h

The *df -h* command displays disk usage in a human-readable format (e.g., GB, MB). Example output shows filesystem name, size, used space, available space, usage percentage, and mount point. It is useful for monitoring storage and ensuring there is enough space. The *-h* flag makes it easier to read compared to bytes-only output. No changes

```
aayusha@DESKTOP-9N9URUN:~/Projects$ df -h
Filesystem      Size   Used  Avail Use% Mounted on
none            3.9G    0  3.9G  0% /usr/lib/modules/6.6.87.2-microsoft-standard-WSL2
none            3.9G  4.0K  3.9G  1% /mnt/wsl
drivers          222G 140G   82G  64% /usr/lib/wsl/drivers
/dev/sdd        1007G 1.6G  955G  1% /
none            3.9G   76K  3.9G  1% /mnt/wslg
none            3.9G    0  3.9G  0% /usr/lib/wsl/lib
rootfs          3.8G  2.7M  3.8G  1% /init
none            3.9G  544K  3.9G  1% /run
none            3.9G    0  3.9G  0% /run/lock
none            3.9G    0  3.9G  0% /run/shm
none            3.9G   76K  3.9G  1% /mnt/wslg/versions.txt
none            3.9G   76K  3.9G  1% /mnt/wslg/doc
C:\             222G 140G   82G  64% /mnt/c
D:\             254G  19G  236G  8% /mnt/d
tmpfs           3.9G  16K  3.9G  1% /run/user/1000
aayusha@DESKTOP-9N9URUN:~/Projects$ |
```

are made to the system by this command. Output lists all mounted filesystems

3. Conclusion

This report showcased a set of frequently used Linux commands, accompanied by clear explanations and actual terminal outputs. Through hands-on practice with these commands, I developed skills in navigating directories, creating and managing files, and efficiently organizing folders. Additionally, the commands enhanced my understanding of monitoring system information, checking disk usage, and viewing active processes. Utilities such as *echo*, *df -h*, *ps*, and *top* strengthened my ability to interact with the system and assess its performance. Overall, becoming familiar with these essential Linux commands has made working with the system more intuitive, efficient, and streamlined.