Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

"Lab Report 01"

[Code No: COMP 307]

Submitted by:

Krishtina Bhatta

Roll no: 09

Submitted to:

Rabina Shrestha

Department of Computer Science and Engineering

Date: 10th December, 2025

INTRODUCTION

1. What is Linux?
   Linux is a powerful family of open-source, Unix-like operating systems built around the Linux kernel, which serves as the core resource manager for computers ranging from smartphones and web servers to desktops and supercomputers. Its defining characteristic is its open-source nature, allowing anyone to freely use, modify and distribute the code, fostering incredible stability, security and flexibility. This flexibility manifests through hundreds of complete systems known as distributions (like Ubuntu and Fedora) which package the kernel with essential utilities, applications and a user interface, providing robust and customizable computing environments suitable for virtual any technical need.

2. The Linux Hierarchical File System.
   The Linux Hierarchical File System (HFS) is a standardized, single-rooted, tree-like directory structure managed entirely under the root directory (/), contrasting sharply with systems that use multiple drive letters. This organization is governed by the Filesystem Hierarchy Standard (FHS), which ensures consistency by defining the purpose of critical directories. For instance, /bin holds essential common programs, /etc contains all system-wide configuration files, /home stores user-specific data and /var is reserved for files that change frequently like system logs. This unified and logically segmented structure allows for easy navigation, efficient resource management and predictable system administration across all Linux distributions.

3. Importance of Linux commands in Operating Systems.
   The importance of Linux commands stems from the fact that they are the primary, most powerful, and most efficient way to interact with the operating system, especially in server and professional environments. While Graphical User Interfaces are convenient for basic tasks, the command-line interface commands offer granular control over the system, enabling system administrators and developers to perform complex operations with precision and speed that are often unavailable or cumbersome in a GUI. Commands are vital for fundamental file management (ls, cp, rm), system administration (chmod, chown, systemctl), and network configuration (ip a, ping, ssh). Most importantly, they are the foundation for automation and scripting, allowing repetitive tasks to be combined, scheduled and executed without manual intervention which is essential for managing large-scale cloud infrastructure and DevOps pipelines. This text-based approach is also highly resource-efficient and reliable for remote access and troubleshooting, making proficiency in Linux commands a fundamental and indispensable skill for managing modern computing environments.

COMMANDS

1. Pwd

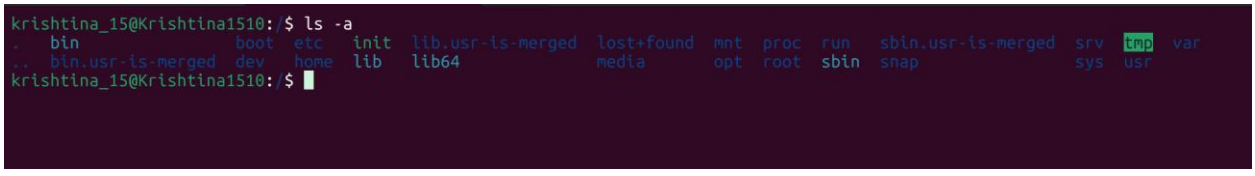
The pwd (Print Working Directory) command displays the absolute path of your current location within the Linux file system, beginning from the root directory (/). It shows the user exactly where they are, which is vital for orientation, especially when navigating deeply nested directories. This command ensures accuracy when executing commands or writing scripts by providing the full, unambiguous path and it can also resolve symbolic links to show the actual physical directory location.

2. ls

The ls command, which stands for list, is one of the most frequently used and fundamental utilities in the Linux command-line interface. Its primary function is to list the contents of a directory, providing an overview of the files and subdirectories present in the specified location. When executed without any arguments, ls defaults to listing the contents of the current working directory.

3. ls -a

The ls -a command forces the listing of all files in a directory, including hidden files and directories whose names begin with a dot (.). This is crucial because many system configuration files and standard directory links (. for the current directory and .. for the parent) are hidden by default. By revealing these dotfiles, ls -a provides a complete and accurate view of the directory's contents, which is vital for system configuration and troubleshooting.



4. ls -l

The ls -l command is a fundamental utility that instructs the Linux shell to list the contents of a directory in the long format. This format is essential because it provides comprehensive, detailed information about every file and directory, going far beyond a simple name list.

```
krishtina_15@Krishtina1510:/$ ls -l
total 2796
lrwxrwxrwx   1 root root       7 Apr 22  2024 bin -> usr/bin
drwxr-xr-x   2 root root    4096 Feb 26  2024 bin.usr-is-merged
drwxr-xr-x   2 root root    4096 Apr 22  2024 boot
drwxr-xr-x  15 root root    3860 Dec 10 11:47 dev
drwxr-xr-x  88 root root    4096 Dec 10 11:47 etc
drwxr-xr-x   3 root root    4096 Dec 10 03:07 home
-rwxr-xr-x   1 root root 2781552 Oct 10 00:22 init
lrwxrwxrwx   1 root root       7 Apr 22  2024 lib -> usr/lib
drwxr-xr-x   2 root root    4096 Apr  8  2024 lib.usr-is-merged
lrwxrwxrwx   1 root root       9 Apr 22  2024 lib64 -> usr/lib64
drwx------   2 root root   16384 Dec 10 03:05 lost+found
drwxr-xr-x   2 root root    4096 Aug  5 16:55 media
drwxr-xr-x   5 root root    4096 Dec 10 03:05 mnt
drwxr-xr-x   2 root root    4096 Aug  5 16:55 opt
dr-xr-xr-x 216 root root       0 Dec 10 11:47 proc
drwx------   3 root root    4096 Aug  5 16:57 root
drwxr-xr-x  19 root root     560 Dec 10 11:47 run
lrwxrwxrwx   1 root root       8 Apr 22  2024 sbin -> usr/sbin
drwxr-xr-x   2 root root    4096 Mar 31  2024 sbin.usr-is-merged
drwxr-xr-x   2 root root    4096 Dec 10 03:05 snap
drwxr-xr-x   2 root root    4096 Aug  5 16:55 srv
dr-xr-xr-x  13 root root       0 Dec 10 11:47 sys
drwxrwxrwt   8 root root    4096 Dec 10 12:13 tmp
drwxr-xr-x  12 root root    4096 Aug  5 16:55 usr
drwxr-xr-x  13 root root    4096 Dec 10 03:05 var
krishtina_15@Krishtina1510:/$
```

5. cd

The cd (change directory) command is the fundamental Linux utility for navigating the file system. It allows users to move their current working location to any other directory. Users can specify an absolute path (starting from /), a relative path (e.g., Documents), or use special shorthand commands like cd .. to move up to the parent directory, or just cd to return instantly to their home directory, making it crucial for terminal efficiency.

```
krishtina_15@Krishtina1510:/$ cd
krishtina_15@Krishtina1510:~$ cd ..
krishtina_15@Krishtina1510:/home$
```

6. mkdir

The mkdir (make directory) command is used to create new folders in the Linux file system. When followed by a name (e.g. mkdir Reports), it creates that directory in the current working location. It can accept multiple names to create several directories simultaneously and crucially, the -p option allows the creation of a full nested directory structure.

```
krishtina_15@Krishtina1510:~$ mkdir Reports
krishtina_15@Krishtina1510:~$ ls
Reports
krishtina_15@Krishtina1510:~$
```

7. rmdir

The rmdir (remove directory) command is used exclusively to delete empty directories from the file system. It will fail with a "Directory not empty" error if the target folder contains any files or subdirectories. The useful -p option allows recursive deletion, removing a directory and its now-empty parent directories in a single command, but its use is limited due to the empty-only restriction.

8. rm

The rm command, shortcut for remove, is the primary and most powerful utility in Linux used for deleting files and directories. Unlike the rmdir command, which is limited to empty directories, rm can delete files, symbolic links, and, when used with the appropriate options, entire directories and all their contents.

9. rm -r

The rm -r command is the essential tool in Linux used for recursively deleting directories and all their contents. Unlike the basic rm command, which is limited to deleting files, the inclusion of the -r (recursive) option signals to the system that the user intends to process a directory.

10. touch

The touch command's primary function is to update the access and modification timestamps of an existing file. However, its most frequent practical use is to create a new, empty file when the specified filename does not yet exist in the directory. This utility is essential for quickly generating placeholder files in the terminal, serving as a quick and simple way to initiate files before adding content later.

```
krishtina_15@Krishtina1510: $ touch new.txt
krishtina_15@Krishtina1510: $ ls
file.txt  graphics.txt  new.txt
krishtina_15@Krishtina1510: $
```

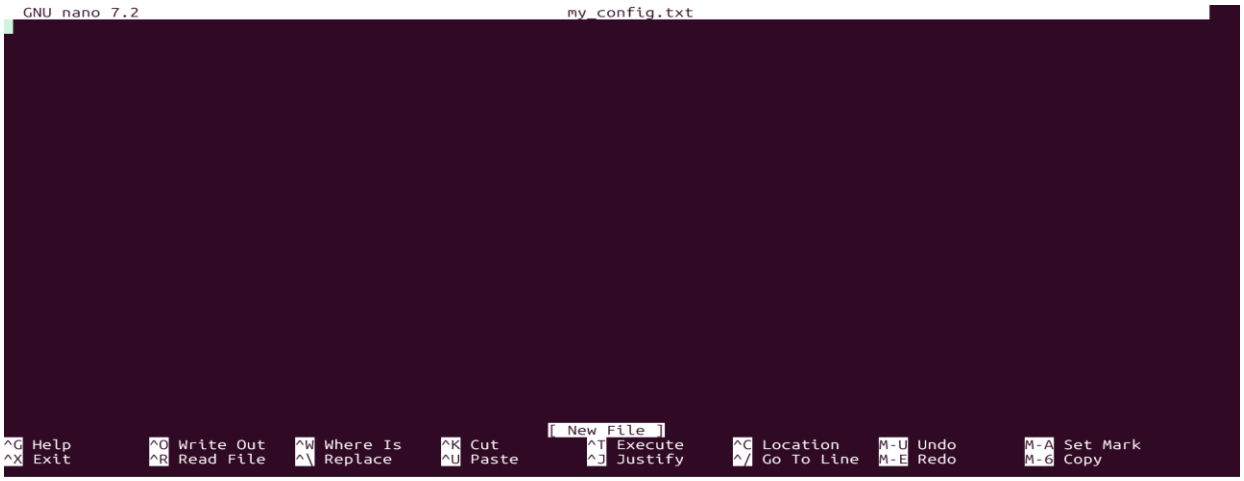
11. cat

The cat command shortform for concatenate is a core Linux utility primarily used to display the contents of text files directly to the terminal screen. It can also concatenate the content of multiple files and redirect the combined output into a new file. Additionally, cat allows for quick file creation or appending content to a file when used with the redirection operators (> or >>) and user input.

```
krishtina_15@Krishtina1510: $ echo "Project Status Report - December 10, 2025" > status.log
krishtina_15@Krishtina1510: $ cat status.log
Project Status Report - December 10, 2025
krishtina_15@Krishtina1510: $
```

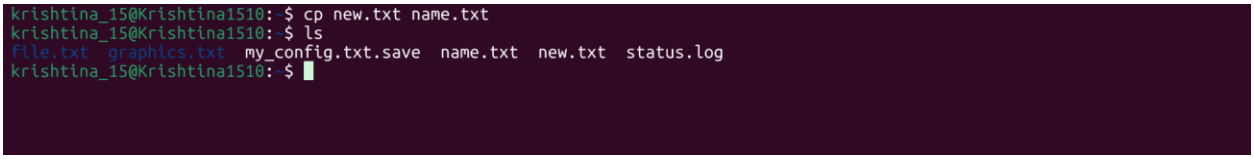
12. navo, vi, jed

Nano, Vi/Vim, and Jed are three of the most important command-line text editors used for configuration, scripting and coding directly within the Linux terminal environment. Nano is the most beginner-friendly editor, featuring a modeless interface where commands like saving and exiting are conveniently listed at the bottom of the screen. In contrast, vi (and its modern version, Vim) is a powerful, standard Unix editor known for its modal operation, requiring users to switch between Command Mode for navigation and actions and Insert Mode for typing while Vim has a steep learning curve, it offers unmatched efficiency for power users and developers. Jed is a feature-rich programmer's editor that typically operates in a modeless fashion and provides strong support for multiple programming languages.



13. cp

The cp command (copy) is a vital Linux utility used to duplicate files and directories from a source to a specified destination. Its basic usage requires specifying the source and the destination (cp source target). To copy an entire directory structure, the -r (recursive) option is required to ensure all contents are copied.



14. mv

The mv command (move) is a fundamental Linux utility used for moving files and directories to a new location, and renaming them. When the destination is a different directory, mv moves the source object without duplicating it. If the destination is simply a new filename in the same location, mv effectively renames the file or directory. Unlike cp, mv does not require a recursive option (-r) to move directories, as it simply updates the file system's record of the object's location.

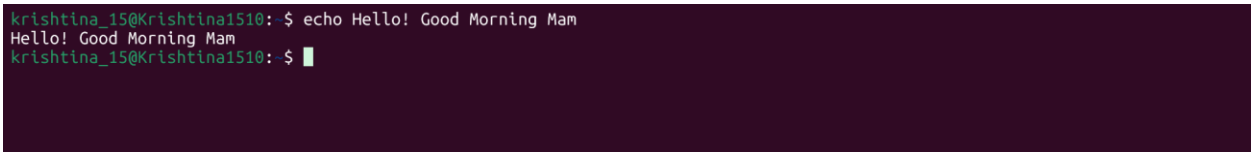
15. locate

The locate command is a very fast and efficient utility in Linux used to quickly find files and directories by name anywhere in the file system. It relies on a pre-built database, typically named mlocate.db, which is created and periodically updated by the system because it searches this indexed database instead of the live file system, locate returns results nearly instantaneously. However, a key limitation is that it cannot find files created since the last database update and for security, it only shows files the user has permission to see.

Since locate is not working in my system I have used the alternate find.

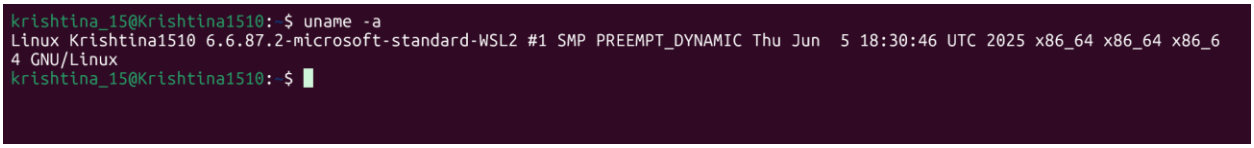
16. echo

The echo command is one of the most basic and frequently used commands in Linux and Unix-like systems. Its primary function is simply to display or print a line of text or a string of characters to standard output, which is typically your terminal window.



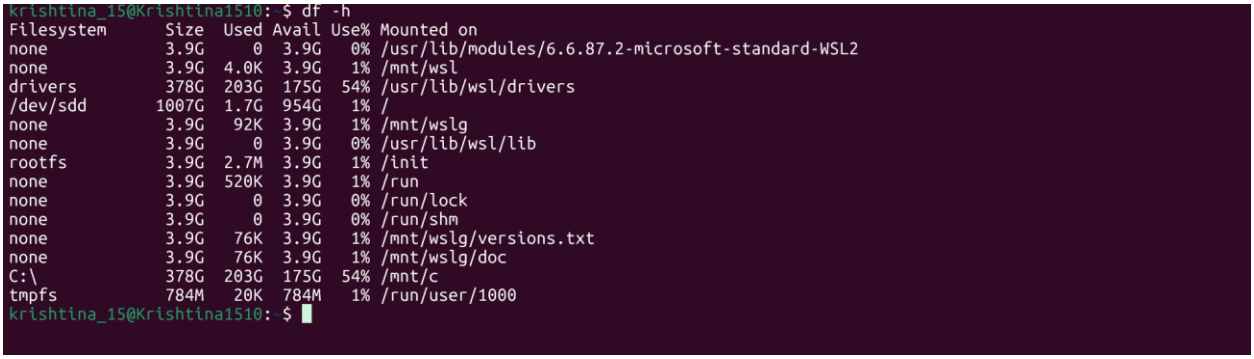
17. uname -a

The uname command displays basic system information, printing only the kernel name (usually Linux) by default. When the -a option is added it prints a comprehensive single string detailing the kernel name, hostname, kernel release and version number, and the hardware architecture (e.g., x86_64). This command is vital for quickly diagnosing system configuration and compatibility issues.



18. df -h

The df command, which stands for disk free, is used to display the amount of available and used disk space on the file systems mounted on your machine. The most common form is df -h. The -h (human-readable) option formats the output sizes to make them easily understandable. This command provides a quick summary of the total disk capacity, how much is currently used, how much is available, and the percentage utilized for each mounted partition.



19. ps -u $USER

The ps command (process status) is used to display information about the currently running processes. When combined with the -u $USER options, it specifically lists all active processes owned by the current logged-in user. This filtered list typically shows essential columns,

including the PID (Process ID), the amount of CPU and memory used, the terminal from which the process was started and the actual COMMAND being executed. This is crucial for system monitoring and identifying resource usage by your own programs.

```
krishtina_15@Krishtina1510: $ ps -u $USER
    PID TTY          TIME CMD
    316 ?        00:00:00 systemd
    317 ?        00:00:00 (sd-pam)
    342 pts/1    00:00:00 bash
   1911 pts/0    00:00:00 bash
   1994 pts/0    00:00:00 ps
krishtina_15@Krishtina1510: $
```

20. top

The top command is a crucial Linux utility that provides a real-time, dynamic view of a system's current performance and running processes. It displays a continuously updated summary of system information, including CPU usage, memory usage, and swap space. Below the summary, it lists individual processes, sorted by default by the process consuming the most CPU time, allowing administrators to quickly identify resource hogs or monitor system health.

```
krishtina_15@Krishtina1510: $ top
top - 14:46:09 up  3:13,  1 user,  load average: 0.00, 0.00, 0.02
Tasks:  24 total,   1 running,  23 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7838.2 total,   7166.1 free,    506.3 used,    320.0 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7331.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0   21720  11972   9156 S   0.3   0.1   0:01.47 systemd
      2 root      20   0    3120   2176   2048 S   0.0   0.0   0:00.00 init-systemd(Ub
      6 root      20   0    3120   1792   1792 S   0.0   0.0   0:00.00 init
     39 root      19  -1   66832  18676  17780 S   0.0   0.2   0:01.49 systemd-journal
     89 root      20   0   25268   6272   4992 S   0.0   0.1   0:02.20 systemd-udevd
    104 systemd+  20   0   21456  12288  10368 S   0.0   0.2   0:00.24 systemd-resolve
    105 systemd+  20   0   91024   7552   6784 S   0.0   0.1   0:00.44 systemd-timesyn
    157 root      20   0    4236   2560   2432 S   0.0   0.0   0:00.04 cron
    158 message+  20   0    9632   5120   4608 S   0.0   0.1   0:00.68 dbus-daemon
    165 root      20   0   17964   8448   7552 S   0.0   0.1   0:00.31 systemd-logind
    167 root      20   0 1756096  13056  10368 S   0.0   0.2   0:00.54 wsl-pro-service
    178 root      20   0    3160   2048   1920 S   0.0   0.0   0:00.01 agetty
    187 syslog    20   0  222508   5248   4480 S   0.0   0.1   0:00.25 rsyslogd
    189 root      20   0    3116   1920   1792 S   0.0   0.0   0:00.01 agetty
    190 root      20   0  107032  22528  13312 S   0.0   0.3   0:00.15 unattended-upgr
    284 root      20   0    6692   4352   3712 S   0.0   0.1   0:00.01 login
    316 krishti+  20   0   20308  11136   9216 S   0.0   0.1   0:00.14 systemd
    317 krishti+  20   0   21152   3516   1792 S   0.0   0.0   0:00.00 (sd-pam)
    342 krishti+  20   0    6056   4992   3456 S   0.0   0.1   0:00.01 bash
   1610 polkitd   20   0  308164   7680   6912 S   0.0   0.1   0:00.09 polkitd
   2014 root      20   0    3124    904    768 S   0.0   0.0   0:00.00 SessionLeader
   2015 root      20   0    3140   1036    896 S   0.0   0.0   0:00.00 Relay(2016)
   2016 krishti+  20   0    6072   5248   3584 S   0.0   0.1   0:00.03 bash
   2029 krishti+  20   0    9276   5504   3456 R   0.0   0.1   0:00.01 top
```

21. chmod

The chmod command (change mode) is used to change the access permissions of a file or directory in Linux. These permissions govern who can read, write, or execute the file (Owner, Group, Others). Permissions are often expressed using a three-digit octal number For example, the command chmod 755 new.txt makes the file executable while setting standard permissions.

```
krishtina_15@Krishtina1510: $ chmod 755 new.txt
krishtina_15@Krishtina1510: $ ls
file.txt  final_report.txt  graphics.txt  my_config.txt.save  name.txt  new.txt  status.log
krishtina_15@Krishtina1510: $
```