Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Lab Report - 1

Linux Commands Lab Report

Course Code: COMP 307

Submitted by:

Apekshya Bhattarai (10)

Submitted to:

Ms. Rabina Shrestha
Department of Computer Science and Engineering

December 10, 2025

1. Introduction

1.1 What is Linux?

Linux is an open-source, Unix like operating system based on the Linux kernel, first released by Linus Torvalds in 1991. Renowned for its stability and security, Linux is widely used in servers, networking, cloud computing, and software development. It is typically distributed as a "distribution" (or distro), which packages the kernel with supporting system software, such as tools and libraries from the GNU Project.

1.2 The Linux Hierarchical File System

The Linux file system is organized hierarchically in a tree-like structure. At the top is the root directory (/), which contains essential system directories like /home, /etc, /bin, and /usr. By default, users' personal files and activities are confined to their assigned subdirectory within /home.

1.3 Importance of Linux Commands

The use of Linux commands is fundamental to leveraging the strengths of the Linux operating system. As a free and open-source platform, Linux is prized for its security, stability, and flexibility. Mastering its commands allows users to fully customize their environment, manage systems efficiently, and tap into a vast ecosystem of software, which is why it dominates server and development landscapes.

2. Linux Commands

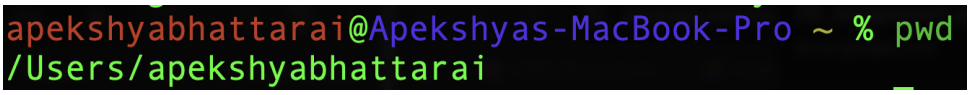
2.1 Command: pwd

Explanation: 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, /Users/apekshyabhattarai. It's useful when navigating through multiple folders.

Output:

```
/Users/apekshyabhattarai
```

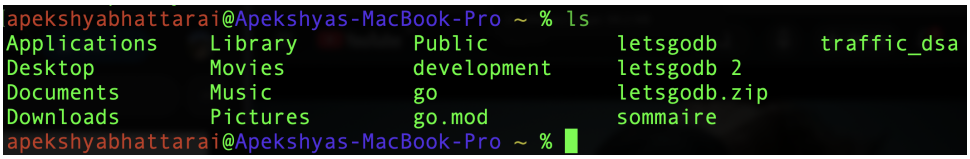
Screenshot:



2.2 Command: ls

Explanation: The ls command is used to list all the files and directories in the current working directory. When the command is executed if nothing was displayed this means 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.

Screenshot:



2.3 Command: ls -a

Explanation: 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 .bash$_h istoryor.config.Inthisexample, runningls-ain/rootshowedthespecialentries.(currentdirec$

Screenshot:

```
apekshyabhattarai@Apekshyas-MacBook-Pro ~ % ls -a
.
..
.CFUserTextEncoding
.DS_Store
.Trash
.bash_history
.cache
.config
.gitconfig
.idlerc
.lesshst
.lldb
.local
.mysql_history
.npm
.oh-my-zsh
.terminfo
.viminfo
```

2.4 Command: ls -l

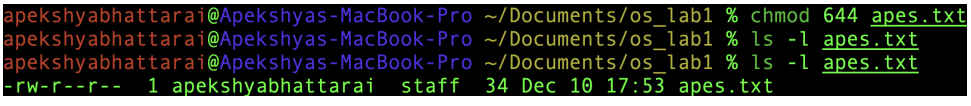
Explanation: The ls -l command displays files and directories in long listing format with detailed information, including:

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

Output:

```
-rw-r--r--  1 apekshyabhattarai  staff  34 Dec 10 17:53 apes.txt
```

Screenshot:

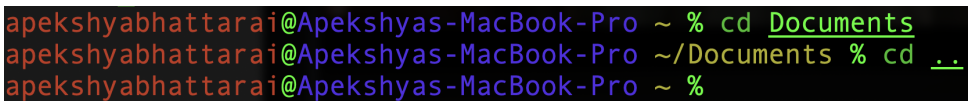


2.5 Command: cd

Explanation: 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.

Screenshot:



2.6 Command: mkdir

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

Example: mkdir os lab1 creates a folder named os lab1 immediately in the current directory.

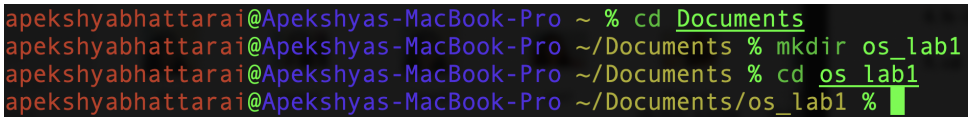
To create nested directories in one command, you can use: mkdir -p folder1/folder2.

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: os lab1

Screenshot:



2.7 Command: rmdir

Explanation: The rmdir command deletes an empty directory.

It cannot delete directories that contain files; attempting to do so will produce an error.

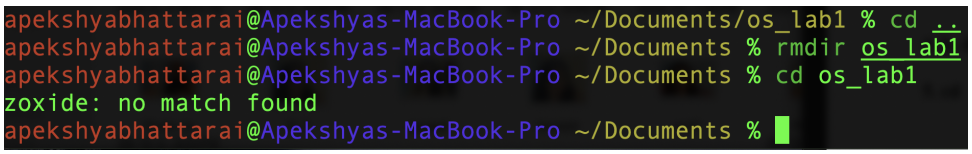
Only directories with no contents can be removed using this command.

It is useful for cleaning up unnecessary empty folders.

Example: rmdir os lab1 deletes the os lab1 folder if it is empty.

If successful, there is no output; if the directory contains files, an error is displayed.

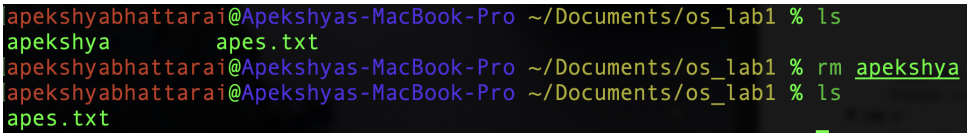
Screenshot:



2.8 Command: rm

Explanation: The rm command is used to delete files permanently.
Example: rm apekshya deletes the file named apekshya.
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.

Screenshot:



2.9 Command: rm -r

Explanation: The rm -r command deletes directories and all their contents recursively.

It can remove folders even if they contain files and subdirectories.

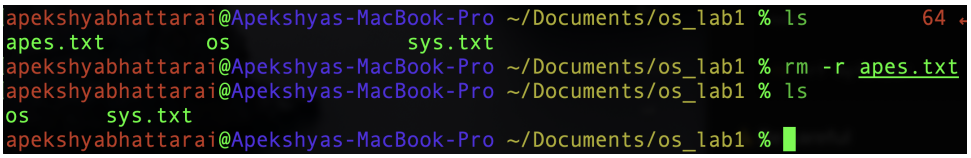
Example: rm -r apes.txt deletes apes.txt along with everything inside it.

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.

Screenshot:



2.10 Command: touch

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

Example: touch apes.txt creates a file named apes.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.

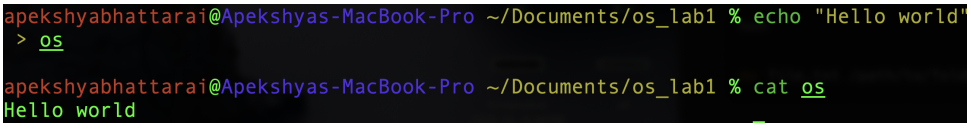
Screenshot:

2.11 Command: echo

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

Example: echo "Hello world" prints Hello world on the screen.

Screenshot:



2.12 Command: cat

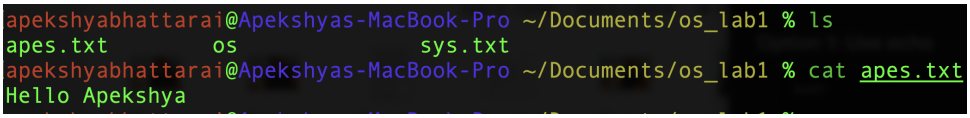
Explanation: The cat command displays the contents of a file in the terminal.

Example: cat notes.txt prints everything in notes.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.

Screenshot:



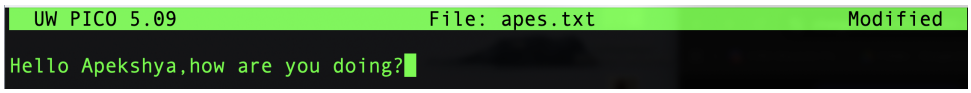
2.13 Command: nano / vi / jed

Explanation: 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:

  ```
  nano apes.txt
  ```

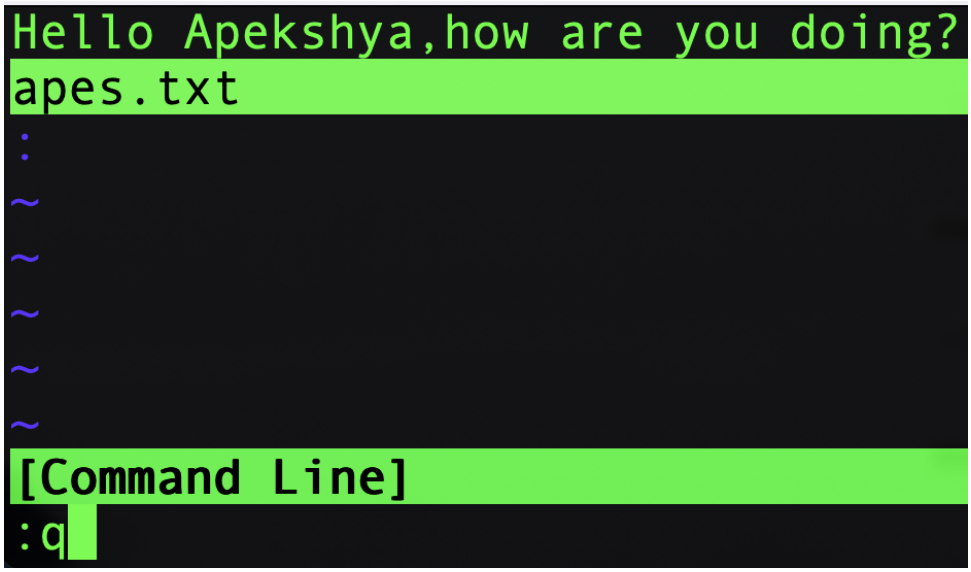
  Screenshot:



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

  ```
  vi apes.txt
  ```

  Screenshot:



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

```
jed apes.txt
```

2.14 Command: cp

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

Example: cp apes.txt sys.txt creates a copy of apes.txt named sys.txt.

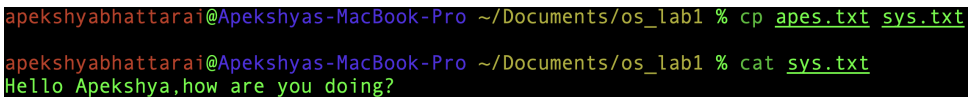
It is useful for making backups or duplicating files.

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.

Screenshot:



2.15 Command: mv

Explanation: The mv command moves or renames files and directories.

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

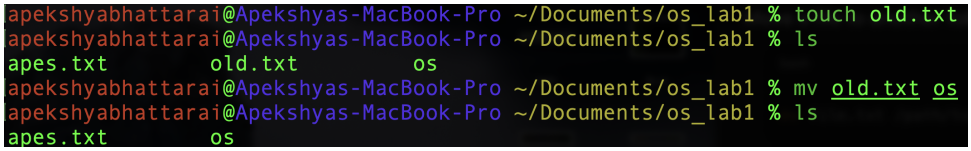
Example: mv file.txt folder/ moves file.txt into the folder folder.

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.

Screenshot:



2.16 Command: locate

Explanation: The locate command searches for files by name quickly using a pre-built database.

Example: locate passwd lists all files containing "passwd" in their path.

The database is updated with updatedb; otherwise, new files may not appear.

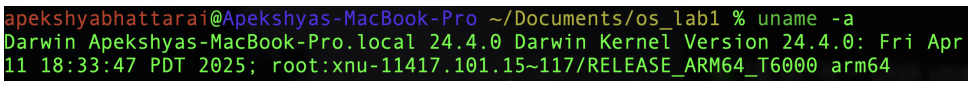
It is much faster than find because it searches a database instead of the filesystem in real-time.

Useful for quickly finding files anywhere on the system.

Output shows the full paths of matching files.

2.17 Command: uname -a

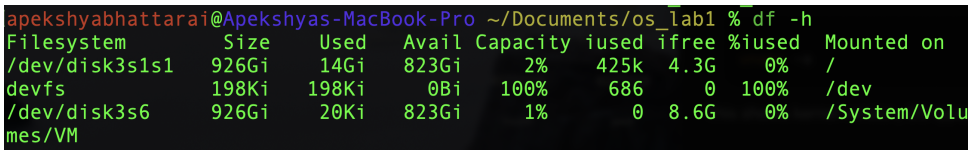
Explanation: Shows system information such as kernel version and architecture.
Screenshot:



2.18 Command: df -h

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

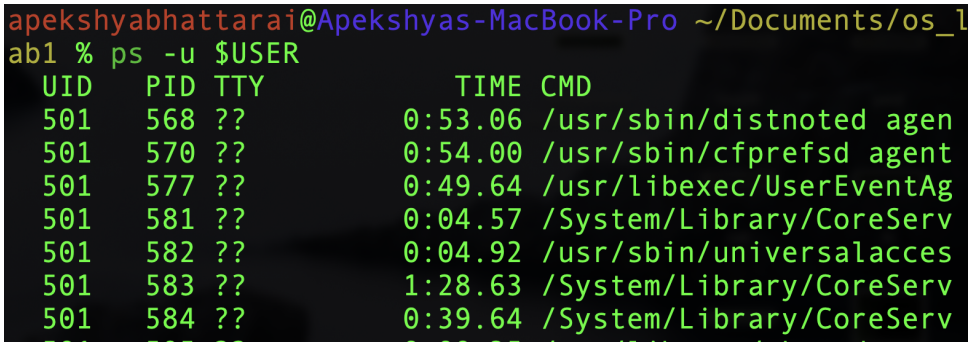
Screenshot:



2.19 Command: ps -u $USER

Explanation: Shows processes running under the current user.

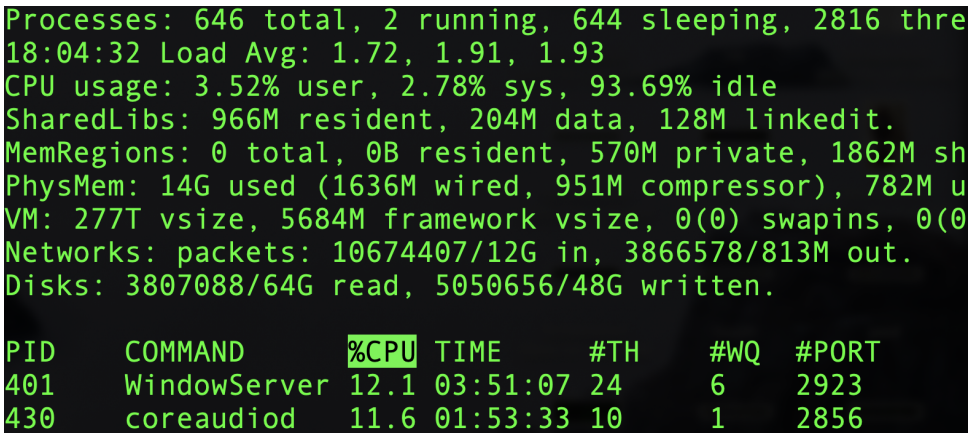
Screenshot:



2.20 Command: top

Explanation: 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.

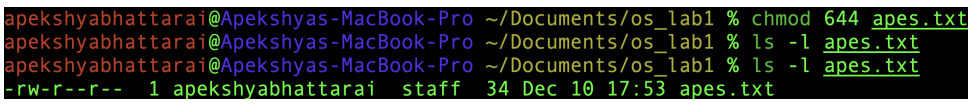
Screenshot:



2.21 Command: chmod

Explanation: Changes file permissions. Example: chmod 755 script.sh.

Screenshot:



3. Conclusion

This report explored a set of frequently used Linux commands, providing explanations and examples from real terminal outputs. Through practicing these commands, I gained skills in navigating directories, creating and managing files, and organizing folders effectively. They also enabled me to monitor system information, check disk usage, and view active processes. Using tools like echo, df -h, ps, chmod, ps -u $USER and top enhanced my ability to interact with the system and assess its performance. Overall, becoming familiar with these essential Linux commands makes working with the system more efficient, faster, and straightforward.