



S.B. JAIN INSTITUTE OF TECHNOLOGY MANAGEMENT & RESEARCH, NAGPUR

Practical 02

Aim: To understand and demonstrate the use of basic commands in different operating systems (Windows, Linux, and UNIX) for managing files, directories, permissions, and user interactions through a terminal or command-line interface.

Name: Akansha Wanjari

USN: CM24004

Semester / Year: 4th/2nd

Academic Session: 2025-26

Date of Performance: / / 26

Date of Submission: / / 26

❖ **Aim:**

To understand and demonstrate the use of basic commands in different operating systems (Windows, Linux, and UNIX) for managing files, directories, permissions, and user interactions through a terminal or command-line interface.

❖ **Objectives:**

- a. Understand and demonstrate the use of basic commands in different operating systems (Windows, Linux, and UNIX).
- b. Manage files and directories using command-line interfaces.
- c. Manage permissions for files and directories.
- d. Perform user interactions through a terminal or command-line interface.
- e. Gain practical experience with OS commands for file system navigation and administration.

❖ **Requirements:**

Hardware Requirements:

- **Processor:** Multi-core CPU, Intel Core i3 (3.0 GHz) or higher
- **RAM:** Minimum 4 GB (8 GB recommended for optimal performance)
- **Storage:** 100 GB HDD or SSD (Solid State Drive) for faster access □
- **Network Interface:** Ethernet or Wi-Fi adapter for connectivity

Software Requirements:

- **Operating System:** Windows 10/11, Linux (Ubuntu 20.04/CentOS 8), UNIX-based OS
- **Command-line Interface:** PowerShell or Command Prompt (Windows), Terminal (Linux/UNIX)
- **Text Editor:** Nano, Vim, or Visual Studio Code for file editing
- **Administrative Privileges:** Superuser (Linux/UNIX) or Administrator (Windows) access

❖ **Theory:**

An Operating System (OS) provides a Command Line Interface (CLI)—a text-based user interface—used to interact with the system's kernel, file system, and applications by typing commands in a shell or terminal. While graphical user

interfaces (GUIs) use icons, the CLI allows for faster, more precise, and scriptable management of system resources, directories, and user permissions.

❖ **Commands:**

1. Display User Manual of a Command

- Functionality: Shows the manual page with details about a command's usage, options, and arguments.
- Syntax: `man <command>`
- Example: `man ls`

2. Change Current Working Directory.

- Functionality: Changes the terminal's current working directory.
- Syntax: `cd <directory-path>`
- Example: `cd /home/user/Documents.`

3. List Contents of the Current Directory.

- Functionality: Lists all files and directories in the current location.
- Syntax: `ls`
- Example: `ls`

4. Read/Modify/Concatenate Text Files.

- Functionality: Displays or manipulates file content.
- Syntax:
 - Read: `cat <filename>`
 - Modify: `'nano <filename>`
 - Concatenate: `cat <file1> <file2> > <outputfile>`

5. Create a New Directory.

- Functionality: Creates a new directory at the specified path.
- Syntax: `mkdir <directory-name>`
- Example: `mkdir newdir`

6. Display Current Working Directory.

- Functionality: Prints the current directory path.
- Syntax: `pwd`
- Example: `pwd`

7. Write Arguments to Standard Output.

- Functionality: Prints the provided string or variables.
- Syntax: echo <arguments>
- Example: echo Hello World

8. Remove a File.

- Functionality: Deletes a specified file.
- Syntax: rm <filename>
- Example: rm file.txt

9. Delete a Directory.

- Functionality: Removes an empty directory.
- Syntax: rmdir <directory-name>
- Example: rmdir olddir

10. Copy a File or Directory.

- Functionality: Copies a file or directory to a destination.
- Syntax: cp <source> <destination>
- Example: cp file.txt backup/

11. Switch to Root User.

- Functionality: Gains root privileges temporarily.
- Syntax: sudo su
- Example: sudo s

12. Move Files or Directories.

- Functionality: Moves or renames files and directories.
- Syntax: mv <source> <destination>
- Example: mv file.txt newdir/

13. Search for a String in a File.

- Functionality: Searches for a specific word or pattern in a file.
- Syntax: grep "<string>" <file>
- Example: grep "error" log.txt

14. Print Top N Lines of a File.

- Functionality: Displays the first N lines of a file.
- Syntax: `head -n <N> <file>`
- Example: `'head -n 10 file.txt'`

15. Print Last N Lines of a File.

- Functionality: Displays the last N lines of a file.
- Syntax: `tail -n <N> <file>`
- Example: `'tail -n 10 file.txt'`

-

16. Remove Read Permission from Owner.

- Functionality: Revokes the owner's read permission for a file.
- Syntax: `chmod u-r <filename>`
- Example: `chmod u-r file.txt`

17. Change Specific Permissions.

- Functionality: Sets or removes specific file permissions.
- Syntax: `chmod u+r,w-x,g+w <filename>`
- Example: `chmod u+r,w-x,g+w file.txt`

18. Add Write Permission to Owner, None to Others.

- Functionality: Allows write access for the owner only.
- Syntax: `chmod u+w,o-rwx <filename>`
- Example: `chmod u+w,o-rwx file.txt`

19. Assign Permissions to Users.

- Functionality: Modifies file access for users, groups, and others.
- Syntax: `chmod u+rwx,g+rx,o+r <filename>`
- Example: `'chmod u+rwx,g+rx,o+r file.txt'`

20. Assign R/W/X to Others.

- Functionality: Gives read, write, and execute permissions to others.
- Syntax: `chmod o+rwx <filename>`

- Example: `chmod o+rw file.txt`

21. Remove All Permissions from All Users.

- Functionality: Clears all permissions on a file.
- Syntax: `'chmod a-rwx <filename>'`
- Example: `'chmod a-rwx file.txt'`

22. Remove Read Permission Using Absolute Mode.

- Functionality: Uses numeric mode to restrict read access.
- Syntax: `chmod 700 <filename>`
- Example: `chmod 700 file.txt`

23. Set R/W for Owner, None for Group/Other.

- Functionality: Assigns permissions in numeric mode.
- Syntax: `chmod 600 <filename>`
- Example: `chmod 600 file.txt'`

- 24. Add Execute for Owner, Read for Group/Others.** - Functionality:
Adds execution and read access. - Syntax: `chmod u+x,g+r,o+r
<filename>`
- Example: `chmod u+x,g+r,o+r file.txt`

25. Add Execute Permission to All Users.

- Functionality: Enables execution by everyone.
- Syntax: `chmod a+x <filename>`
- Example: `chmod a+x script.sh` Output:

❖ **Conclusion:.**

students will have gained hands-on experience with basic command-line operations in Windows, Linux, and UNIX. They will be able to manage files and directories, set permissions, and interact with the system via the terminal, thereby strengthening their understanding of OS command-line interfaces and their practical usage in system administration and file management.

```
student@student-BY-OEM:~$ head file.txt
head: cannot open 'file.txt' for reading: No such file or directory
student@student-BY-OEM:~$ ps
  PID TTY          TIME CMD
 3492 pts/0    00:00:00 ps
 5878 pts/0    00:00:00 bash
student@student-BY-OEM:~$ PrtSc
PrtSc: command not found
student@student-BY-OEM:~$ shift + PrtSc
bash: shift: +: numeric argument required
student@student-BY-OEM:~$ rm file.txt
rm: cannot open 'file.txt' for reading: No such file or directory
student@student-BY-OEM:~$ cp file.txt backup.txt
cp: file.txt: No such file or directory
student@student-BY-OEM:~$ mv old.txt new.txt
mv: cannot open 'old.txt' for reading: No such file or directory
student@student-BY-OEM:~$ cat file.txt
cat: file.txt: No such file or directory
student@student-BY-OEM:~$ ls
total 12
drwxr-xr-x 3 student student 4096 Nov 12 12:34 .
drwxr-xr-x 3 student student 4096 Nov 12 12:34 ..
-rw-r--r-- 1 student student    0 Nov 12 12:34 file.txt
-rw-r--r-- 1 student student    0 Nov 12 12:34 new.txt
student@student-BY-OEM:~$ cd /home/student
student@student-BY-OEM:~/home/student$ ls
total 12
drwxr-xr-x 3 student student 4096 Nov 12 12:34 .
drwxr-xr-x 3 student student 4096 Nov 12 12:34 ..
-rw-r--r-- 1 student student    0 Nov 12 12:34 file.txt
-rw-r--r-- 1 student student    0 Nov 12 12:34 new.txt
student@student-BY-OEM:~/home/student$ cd Downloads
student@student-BY-OEM:~/home/student/Downloads$ ls
total 12
drwxr-xr-x 2 student student 4096 Nov 12 12:34 .
drwxr-xr-x 3 student student 4096 Nov 12 12:34 ..
-rw-r--r-- 1 student student    0 Nov 12 12:34 file.txt
-rw-r--r-- 1 student student    0 Nov 12 12:34 new.txt
student@student-BY-OEM:~/home/student/Downloads$ cp file.txt backup.txt
cp: file.txt: No such file or directory
student@student-BY-OEM:~/home/student/Downloads$ mv old.txt new.txt
mv: cannot open 'old.txt' for reading: No such file or directory
student@student-BY-OEM:~/home/student/Downloads$ cat file.txt
cat: file.txt: No such file or directory
student@student-BY-OEM:~/home/student/Downloads$ ls
total 12
drwxr-xr-x 2 student student 4096 Nov 12 12:34 .
drwxr-xr-x 3 student student 4096 Nov 12 12:34 ..
-rw-r--r-- 1 student student    0 Nov 12 12:34 file.txt
-rw-r--r-- 1 student student    0 Nov 12 12:34 new.txt
student@student-BY-OEM:~/home/student/Downloads$
```

❖ Discussion Questions:

1. How do file permission models differ between Windows and UNIX/Linux, and what commands are used to modify them in each system?
2. What are the equivalent commands for creating and navigating directories in Windows (Command Prompt) and Linux/UNIX (terminal), and how do their syntaxes compare?
3. Explain how user management via the command line affects file access permissions in UNIX/Linux versus Windows
4. What are the key differences in handling file paths (absolute vs. relative) across Windows, Linux, and UNIX command-line environments?
5. How would you perform batch file operations (e.g., copying or deleting multiple files) using scripts in Windows (batch files) compared to shell scripts in Linux/UNIX?

❖ **References:**

<https://ubuntu.com/tutorials/command-line-for-beginners#1-overview>

<https://www.geeksforgeeks.org/25-basic-ubuntu-commands/>

Date: / / **2026**

Signature

Course Coordinator

B.Tech CSE(AIML)

Sem: 4 / 2025-26