



## **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:** Puja D. Jadhav

**USN:** CM24037

**Semester / Year:** 4<sup>th</sup>/2<sup>nd</sup>

**Academic Session:** 2025-26

**Date of Performance:**

**Date of Submission:**

*Department of Emerging Technologies, SBJITMR, Nagpur.*

- ❖ **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.

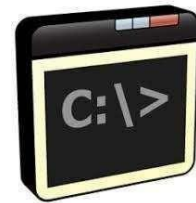
❖ **Objectives:**

1. To learn and practice fundamental command-line operations for file and directory management.
2. To explore and utilize user and permission management commands effectively.
3. To enhance system administration skills by working with commands across different operating systems.

❖ **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:**

In system administration, command-line interfaces (CLI) are essential tools for managing and interacting with operating systems like Windows, Linux, and UNIX. Commands allow users to perform various tasks such as navigating directories, managing files, controlling permissions, and monitoring system performance. Each operating system provides a set of built-in commands, such as 'man', 'ls', 'cd', 'mkdir', and 'chmod', to facilitate efficient system management. Understanding these commands and their syntax is crucial for automating tasks, enhancing security, and ensuring optimal system functionality. This practical aims to develop foundational skills in executing and applying basic commands across different platforms.

❖ **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+rwx 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:

```
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 bash
5878 pts/0      00:00:00 ps
student@student-BY-OEM:~$ prtsc
prtsc: command not found
student@student-BY-OEM:~$ PrtSc
PrtSc: command not found
student@student-BY-OEM:~$ shift + PrtSc
bash: shift: ++: numeric argument required
student@student-BY-OEM:~$ ps
PID TTY          TIME CMD
3492 pts/0      00:00:00 bash
5919 pts/0      00:00:00 ps
student@student-BY-OEM:~$ kill 1234
bash: kill: (1234) - No such process
student@student-BY-OEM:~$ rm file.txt
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 bash
5967 pts/0      00:00:00 ps
student@student-BY-OEM:~$
```

```
student@student-BY-OEM:~$ ls(1)
NAME
ls - list directory contents
SYNOPSIS
ls [OPTION]... [FILE]...
DESCRIPTION
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is
specified.
Mandatory arguments to long options are mandatory for short
options too.
-a, --all
do not ignore entries starting with .
-A, --almost-all
do not list implied . and ..
--author
```

```
student@student-BY-OEM:~$ cp file.txt backup.txt
student@student-BY-OEM:~$ mv old.txt new.txt
mv: cannot move 'old.txt' to 'new.txt': No such file or directory
student@student-BY-OEM:~$ rm file.txt
rm: cannot remove 'file.txt': No such file or directory
student@student-BY-OEM:~$ cat file.txt
cat: file.txt: No such file or directory
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 bash
5878 pts/0      00:00:00 ps
student@student-BY-OEM:~$ prtsc
prtsc: command not found
student@student-BY-OEM:~$ PrtSc
PrtSc: command not found
student@student-BY-OEM:~$ shift + PrtSc
bash: shift: ++: numeric argument required
student@student-BY-OEM:~$ ps
PID TTY          TIME CMD
3492 pts/0      00:00:00 bash
5919 pts/0      00:00:00 ps
student@student-BY-OEM:~$ kill 1234
bash: kill: (1234) - No such process
student@student-BY-OEM:~$ rm file.txt
student@student-BY-OEM:~$ cat file.txt
cat: file.txt: No such file or directory
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 bash
5967 pts/0      00:00:00 ps
student@student-BY-OEM:~$
```

```
student@student-BY-OEM:~$ pwd
/home/student
student@student-BY-OEM:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
student@student-BY-OEM:~$ cd Document
bash: cd: Document: No such file or directory
student@student-BY-OEM:~$ mkdir test
student@student-BY-OEM:~$ touch file.txt
student@student-BY-OEM:~$ cp file.txt backup.txt
student@student-BY-OEM:~$ mv old.txt new.txt
mv: cannot move 'old.txt' to 'new.txt': No such file or directory
student@student-BY-OEM:~$ rm file.txt
rm: cannot remove 'file.txt': No such file or directory
student@student-BY-OEM:~$ cat file.txt
cat: file.txt: No such file or directory
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 bash
5878 pts/0      00:00:00 ps
student@student-BY-OEM:~$
```

❖ **Conclusion:** In conclusion, understanding and using essential operating system commands like ‘ls’, ‘cd’, ‘cp’, ‘mv’, and ‘chmod’ enables efficient file management, navigation, and permission control. Tools like ‘grep’, ‘head’, and ‘tail’ enhance data processing. Mastery of these commands improves system administration, task automation, and overall system security and performance.

❖ **Discussion Questions:**

1. What is the significance of the pwd command in a Linux environment?
2. Explain the function of the cp command and its common options.
3. How does chmod 700 affect file permissions, and what does each digit represent?
4. Describe the difference between head and tail commands in Linux.
5. What is the purpose of the grep command, and how is it used with regular expressions?

❖ **References:**

<https://ubuntu.com/tutorials/command-line-for-beginners#1-overview>  
<https://www.geeksforgeeks.org/25-basic-ubuntu-commands/>

Date:   /  /  

---

**Signature**

Course Coordinator  
B.Tech CSE(AIML)  
Sem: 4 / 2025-26