

# Operating Systems Laboratory (CSE 4510)

## Week 2

### Topic: Basic shell commands and tools.

#### PWD

Command	Purpose	Example / Result
pwd	Shows current working directory	pwd Result: /home/enamul

#### LS

Command	Purpose	Example / Result
ls	List files and directories	ls Result: Desktop Documents Downloads
ls -l	Long listing with details	ls -l Result: -rw-r--r-- 1 user user 1200 file.txt
ls -a	Show hidden files	ls -a Result: ... .bashrc folder
ls -lh	Human-readable sizes	ls -lh Result: -rw-r-- 1 user 1.2K file.txt
ls -R	Recursive listing	ls -R folder/

#### MKDIR

Command	Purpose	Example / Result
mkdir lab	Create a directory	mkdir lab Result: folder 'lab' created
mkdir -p project/week2	Create nested directories	mkdir -p project/week2

#### RM

Command	Purpose	Example / Result
rm file.txt	Remove a file	rm file.txt Result: file deleted
rm -r folder	Delete directory recursively	rm -r folder
rm -f file.txt	Force delete without prompt	rm -f file.txt

#### GREP

Command	Purpose	Example / Result
grep 'hello' file.txt	Search for a pattern in a file	grep 'hello' file.txt Result: hello world
grep -i 'error' log.txt	Case-insensitive search	grep -i error log.txt
grep -n 'main' code.c	Show line numbers	3: int main() {
grep -r 'login' /var/log	Recursive search	log/auth.log: login success
grep -c 'student' data.txt	Count matches	Result: 5

#### FIND

Command	Purpose	Example / Result
find . -name '*.txt'	Find files by name	Result: ./notes.txt
find /home -type f -name 'a.txt'	Find specific file	/home/user/a.txt
find . -type d -name 'backup'	Find directories	./backup
find . -size +1M	Find files larger than 1MB	Result: bigfile.iso
find . -name '*.log' -exec rm {} \;	Run command on results	Deletes all .log files

## HEAD

Command	Purpose	Example / Result
head file.txt	Show first 10 lines	Result: First 10 lines displayed
head -n 5 file.txt	Show first 5 lines	Result: Line1–Line5
head -c 20 file.txt	Show first 20 bytes	Result: first 20 characters

## TAIL

Command	Purpose	Example / Result
tail file.txt	Show last 10 lines	Result: last 10 lines
tail -n 20 file.txt	Show last 20 lines	Result: last 20 lines
tail -f /var/log/syslog	Follow live logs	Continuously updates

## MAN

Command	Purpose	Example / Result
man ls	Show manual for a command	Result: LS documentation
man grep	Manual for grep	Search explanation

## Linux CHMOD – Permission Changing Guide

The 'chmod' command in Linux is used to change file and folder permissions. It controls who can read, write, or execute a file. Linux has three types of users:

- Owner (u)
- Group (g)
- Others (o)

And three types of permissions:

Read (r = 4)

Write (w = 2)

Execute (x = 1)

Permissions can be set using either symbolic mode (u+r, g-w, etc.) or numeric mode (755, 644, etc.). Where the numeric value in each place can be an octal number. And its binary represents the permissions, like 0 means permission not given, 1 means permission given.

Pattern: chmod WhoWhatWhich filename.ext [who(owner)→u/g/o, what(add/remove)→-/+/=, Which(permission type)→r/w/x]

### Basic CHMOD Commands

Command	Purpose / Meaning	Example / Result
chmod 755 file.sh	Owner: rwx, Group: r-x, Others: r-x	Gives execute permission to owner & read/execute to others.
chmod 644 file.txt	Owner: rw-, Group/Others: r--	Standard permission for text files.
chmod 700 secret.txt	Owner only: rwx	Makes file private to the owner.
chmod 600 key.pem	Owner: rw-, Group/Others: ---	Used for private keys & sensitive files.

## Symbolic Mode Commands

Command	Purpose / Meaning	Example / Result
chmod u+x script.sh	Add execute for owner	Owner can now run the script.
chmod g-w report.txt	Remove write permission from group	Group loses ability to edit file.
chmod o+r notes.txt	Add read permission for others	Anyone can read the file.
chmod a+x run.sh	Add execute to all users	Everyone can run the script.

## Folder Permission Commands

Command	Purpose / Meaning	Example / Result
chmod -R 755 project/	Apply recursively to all subfolders/files	Useful for web/server directories.
chmod 777 shared/	Full rights for everyone	⚠ Not recommended. Unsafe permission.

## Pipe (|) in Linux?

A **pipe** is used to **send the output of one command as the input of another command**.

Pipe connects two commands the left command *produces*, the right command *processes*.

Command 1 → Command 2

(output)            (input)

## basic Pipe Usage

Command	Purpose	Example / Result
ls   head	Send output of ls to head (show first 10 items)	Example: ls   head Result: file1 file2 file3 ...
cat file.txt   tail	Show last 10 lines of a file using pipe	Example: cat file.txt   tail Result: (last 10 lines displayed)

Basic Pipe Usage

Command	Purpose	Example / Result
ls   head	Send output of ls to head (show first 10 items)	Example: ls   head Result: file1 file2 file3 ...
cat file.txt   tail	Show last 10 lines of a file using pipe	Example: cat file.txt   tail Result: (last 10 lines displayed)

Using head and tail Together

Command	Purpose	Example / Result
head -20 file.txt   tail -5	Show lines 16–20 from a file	Example: head -20 file.txt   tail -5 Result: Line16 Line17 Line18 Line19 Line20

Extracting Specific Line Number

Command	Purpose	Example / Result
head -n 10 file.txt   tail -n 1	Show exactly line 10	Example: head -n 10 file.txt   tail -n 1 Result: (This is line 10)
head -n 50 data.txt   tail -n 1	Show exactly line 50	Example: head -n 50 data.txt   tail -n 1 Result: (This is line 50)

Pipe with grep

Command	Purpose	Example / Result
grep 'error' log.txt   head	Show first 10 matching 'error' lines	Example: grep 'error' log.txt   head Result: First 10 matched lines
grep 'error' log.txt   head -3   tail -1	Show 3rd matching error line	Example: grep 'error' log.txt   head -3   tail -1 Result: (3rd matched line)

## Multiple Pipe Chaining

Command	Purpose	Example / Result
<code>cat words.txt   sort   uniq   head -5</code>	Sort → remove duplicates → show first 5	Example: <code>cat words.txt   sort   uniq   head -5</code> Result: word1 word2 word3 word4 word5

## Linux Practice Problems

### Problem 1: Directory Navigation & File Creation

- Go to your home directory.
- Create a folder named 'week2\_lab'.
- Inside it, create another folder named 'notes'.
- Verify the folder structure using `ls -R`.

### Problem 2: Create & Edit Files

- Inside `week2_lab/notes`, create a file named 'intro.txt' using nano.
- Write 5 lines about 'What I learned today in Linux'.
- Save and exit.
- Display the content of the file using `cat`.

### Problem 3: Viewing Specific Lines

- Show the first 2 lines of `intro.txt` using `head`.
- Show the last line using `tail`.
- Show line 3 only using: `head -n 3 intro.txt | tail -n 1`

### Problem 4: File Permissions (chmod)

- Check file permissions of `intro.txt` using `ls -l`.
- Remove write permission for group and others.
- Add execute permission for only the owner.
- Verify updated permissions using `ls -l`.

### Problem 5: Searching Inside Files (grep)

- Create a file 'students.txt' with multiple names including duplicates.

- Find all lines containing 'Hasan'.
- Count how many times 'Hasan' appears.
- Show the first 2 'Hasan' results using pipes.
- Show the last 'Hasan' result using tail.

### **Problem 6: Finding Files (find)**

- Find all .txt files inside your home directory.
- Find a file named intro.txt anywhere in home directory.
- Find all directories named 'notes'.

### **Problem 7: Deleting & Managing Files**

- Copy intro.txt into the parent folder.
- Rename the copy to intro\_backup.txt.
- Delete intro\_backup.txt.
- Delete the notes folder completely using rm -r notes.

### **Problem 8: Pipes with head & tail**

- Show first 10 lines of system.log using a pipe.
- Show last 20 lines using a pipe.
- Show line number 50 using: head -n 50 system.log | tail -n 1
- Show lines containing the word 'error'.
- Show the first 3 error lines using: grep 'error' system.log | head -3

### **Problem 9: Mixed Command Challenge**

- Show only .txt files from the challenge folder.
- Show all files sorted by size.
- Find lines in data1.log containing 'failed'.
- Extract line 12 from data2.log.
- Remove execute permission from 'secret'.
- Add read permission to everyone.

### **Problem 10: Mini Project (Everything Combined)**

- Create a directory 'my\_project'.
- Create a file 'info.txt' using nano and write 8 lines.
- Extract lines 3–6 using pipes.
- Search for the word 'Linux' inside info.txt.
- Copy info.txt to info\_backup.txt.

- Change permissions of info\_backup.txt to: rwxr-----
- Confirm the permission.
- Find info\_backup.txt from the home directory using find.