## Files & Directories (Core)

Linux Commands Course · Section 2

#### Goal

Understand how to create, view, modify, and organize files and directories in Linux.

You'll learn to handle files safely, read them efficiently, and manage structure with precision.

## **Everything is a File**

In Linux, almost everything is treated as a **file** — whether it's a document, folder, device, or socket.

- Regular files → data you create (.txt, .py, .jpg)
   Directories → special files that store file lists
- Devices → /dev/sda, /dev/null
- Processes → /proc/<pid>
- Links → alternate names or shortcuts to files

## **Creating Files — touch**

touch cr	eates	an	empty	file	if	it	doesn'	t	exist.
----------	-------	----	-------	------	----	----	--------	---	--------

touch notes.txt

If the file exists, touch updates its modification timestamp.

You can create multiple files at once:

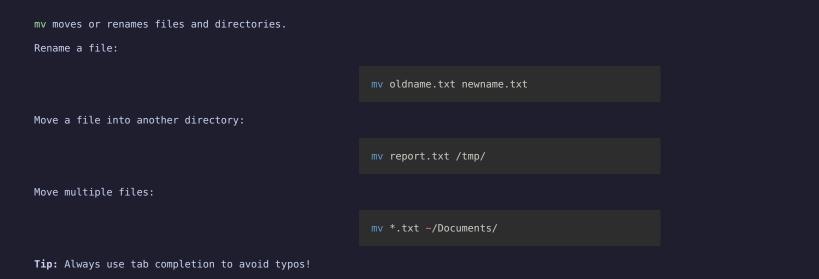
touch a.txt b.txt c.txt

## Reading Files — cat, less, nl

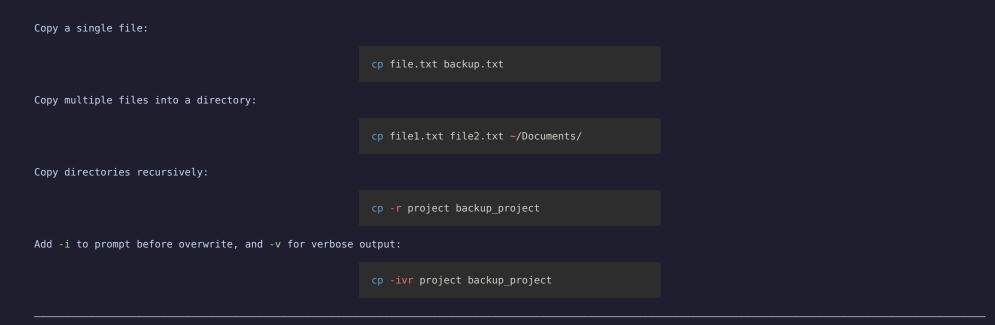
## **Previewing Files — head and tail**



## Renaming & Moving — mv



## Copying Files — cp



# Deleting Files & Folders - rm, rmdir

Delete a file:		
	rm file.txt	
Delete multiple files:		
	rm *.log	
Remove a directory recursively (careful!):		
	rm -r old_project	
Ask before each deletion:		
	rm -ri old_project	
Remove empty directories safely:		
	rmdir emptydir	

# Creating Directories - mkdir

Create one directory:		
	mkdir projects	
Create nested directories in one go:		
	mkdir -p projects/python/scripts	
-p ensures parent folders are created if missing.		

#### Inspecting File Metadata - stat

stat displays detailed information about a file.

stat notes.txt

Example output:

File: notes.txt
Size: 4096 Blocks: 8 IO Block: 4096 regular file
Device: 802h/2050d Inode: 1234567 Links: 1
Access: (0644/-rw-r--r--) Uid: (1000/student) Gid: (1000/student)
Access, Modify, Change times...

Shows size, type, permissions, timestamps, and inode (unique identifier).

## Detecting File Type — file

Check what kind of data a file contains.

file /bin/bash file photo.jpg file script.sh

#### Output examples:

- ELF 64-bit executable (for programs)
- JPEG image data ASCII text

It's a quick way to understand what a file really is, regardless of its extension.

## Links — Hard vs Symbolic



## Differences Between Link Types

Feature	Hard Link	Symbolic Link
Points to Works across filesystems Affected if original deleted Shown in ls -l	file's inode (real data)  X  stays (until inode reused) same inode number	file path (name) ✓ breaks (dangling link) with → target path

Use symbolic links for convenience and hard links for redundancy.

#### **Safety Tips**

- Use -i (interactive) with cp, mv, and rm while learning.
  Always double-check paths before using rm -r.

- Use less instead of cat for large files.
  For log monitoring, combine tail -f with grep.

#### Example:

tail -f /var/log/syslog | grep "error"

#### Recap

- Create files → touch
- **Read** → cat, less, nl, head, tail -f
- Modify / Move → cp, mv, rm
   Directories → mkdir -p, rmdir
   Inspect → stat, file
   Links → ln, ln -s

These are your daily drivers for file management in Linux.

#### **Practice**

- 1. Create a directory named lab.
- Inside it, make an empty file report.txt.
   View it with cat, then less.

- 4. Copy it to backup/ and rename it.5. Create a symbolic link to it called latest\_report.6. Delete the original what happens to the symlink?
- 7. Inspect the file type using file.
  8. Check file details with stat.

## Next Up

Permissions & Ownership (Core) — understanding who can do what with files.