File Viewing & Comparison Commands in Linux

1️⃣ head – View Beginning of File

🔹 Definition

The `head` command is used to display the **first 10 lines** of a file by default.

🔹 Syntax

```
head file_name
```

🔹 Examples

✅ Display first 10 lines (default)

```
head file.txt
```

👉 Shows first 10 lines of `file.txt`

✅ Display first 6 lines

```
head -n 6 file.txt
```

👉 Shows first 6 lines

🔹 When to Use head?

- To quickly check beginning of log file
- To verify file header
- To preview large files

2️⃣ tail – View End of File

🔹 Definition

The `tail` command displays the **last 10 lines** of a file by default.

🔹 Syntax

```
tail file_name
```

🔹 Examples

✅ Display last 10 lines (default)

```
tail file.txt
```

✅ Display last 4 lines

```
tail -n 4 file.txt
```

🔹 Live Log Monitoring (Very Important)

✅ Monitor Live Logs

```
tail -f access.log
```

👉 `-f` means **follow**

This command:

- Shows last 10 lines
- Continues to display new lines as they are added
- Mostly used for monitoring logs

Example usage:

- Apache logs
- Nginx logs
- Application logs

To stop live monitoring:

```
Press Ctrl + C
```

3️⃣ less – View File Page by Page (Scrollable)

🔹 Definition

The `less` command displays file content **one screen at a time**.

It allows:

- Forward movement

- Backward movement
- Searching inside file

🔹 Syntax

```
less file.txt
```

🔹 Navigation Inside less

Key	Action
Space	Next page
b	Previous page
/word	Search word
q	Quit

🔹 Why less is Better?

- Can scroll forward and backward
- Suitable for large files
- Does not load entire file into memory

4️⃣ more – View File Page by Page (Basic Version)

🔹 Definition

The `more` command displays content page by page with percentage read.

🔹 Syntax

```
more file.txt
```

🔹 Navigation

Key	Action
Space	Next page
q	Quit

🔹 Difference Between less and more

Feature	less	more
Forward scroll	✅ Yes	✅ Yes
Backward scroll	✅ Yes	❌ No
Search	✅ Yes	Limited
Recommended	✅ Yes	❌ Basic use

👉 In real-time usage, **less is preferred over more**

5️⃣ diff – Compare Two Files or Directories

🔹 Definition

The `diff` command compares two files (or directories) and shows the differences **line by line**.

It tells:

- What lines were **added (a)**
- What lines were **deleted (d)**
- What lines were **changed (c)**

✅ Basic Syntax

- `diff file1 file2`

📌 Example

Step 1: Create two files

file1.txt

- `Linux`
- `DevOps`
- `Cloud`
- `Docker`

file2.txt

- `Linux`
- `DevOps`
- `AWS`
- `Docker`
- `Kubernetes`

Step 2: Run diff

- `diff file1.txt file2.txt`

🔍 Output Explanation

You may see something like:

- `3c3`
- `< Cloud`
- `---`
- `> AWS`
- `4a5`
- `> Kubernetes`

Let's understand this 👇

🔎 Meaning of Output

`3c3`

- Line 3 in file1 changed to line 3 in file2
- `c` means **change**
- `< Cloud` `(from file1)`
- `---`
- `> AWS` `(from file2)`

4a5

- After line 4 in file1, line 5 added in file2
- `a` means **add**
- `> Kubernetes`

🔷 diff Symbols Meaning

Symbol	Meaning
`a`	Add
`c`	Change
`d`	Delete
`<`	Line from first file
`>`	Line from second file

🔥 More Useful Options

Show difference side by side

- `diff -y file1.txt file2.txt`

Ignore case

`diff -i file1.txt file2.txt`

The `diff` command is not only for files — it can also compare **entire directories**.

🔹 Basic Syntax for Directory Comparison

```
diff -r dir1 dir2
```

👉 `-r` means **recursive** (compare all subdirectories and files inside).

📌 Example

Step 1: Create Two Directories

```
dir1/
 ├── file1.txt
 ├── file2.txt

dir2/
 ├── file1.txt
 ├── file3.txt
```

Step 2: Run Command

```
diff -r dir1 dir2
```

🔍 Example Output

```
Only in dir1: file2.txt

Only in dir2: file3.txt

diff dir1/file1.txt dir2/file1.txt
```

6️⃣ vimdiff / vim -d

`vimdiff` is used to **compare two or more files side-by-side inside Vim** and highlight differences.

It is very useful for:

- Comparing configuration files
- Checking code changes
- Reviewing deployments
- Merging changes manually

🔹 1️⃣ Basic Syntax

Compare two files:

```
vimdiff file1.txt file2.txt
```

You can also use:

```
vim -d file1.txt file2.txt
```

Both do the same thing.

🔹 2️⃣ What Happens When You Open vimdiff?

- Files open side-by-side
- Differences are highlighted
- You can navigate between changes
- You can edit and merge content

🔹 3️⃣ Important Navigation Commands

Command	Action
`]c`	Go to next difference
`[c`	Go to previous difference
`Ctrl + w w`	Switch window
`Ctrl + w h/j/k/l`	Move between windows

7️⃣ wc – Word Count

🔹 Definition

The `wc` command displays:

- Line count
- Word count
- Character count

🔹 Syntax

```
wc file.txt
```

🔹 Example

```
wc file.txt
```

Output format:

```
10  50  300 file.txt
```

Meaning:

Number	Meaning
10	Lines
50	Words
300	Characters

🔹 Useful Options

✅ Count only lines

```
wc -l file.txt
```

✅ Count only words

```
wc -w file.txt
```

✅ Count only characters

```
wc -c file.txt
```

8️⃣ Commands Used to Read Files in Linux

Linux provides multiple commands to read files:

- `cat`
- `tac`
- `vim`
- `head`
- `tail`
- `less`
- `more`

📌 When to Use Which Command?

Situation	Command

Small file	cat
Reverse reading	tac
Large file viewing	less
See first few lines	head
See last few lines	tail
Monitor logs	tail -f
Compare files	diff
Count lines/words	wc

📖 Summary

- `head` → First lines
- `tail` → Last lines
- `tail -f` → Live logs
- `less` → Scrollable view
- `more` → Basic page view
- `diff` → Compare files
- `wc` → Count lines, words, characters
