# CLI Basics: Commands & Navigation

Session 4

# CLI Basics: Commands & Navigation

## Session 4

**Objectives:**

- Understand the CLI structure.
- Navigate the filesystem using basic commands.
- List files and directories.
- Create, move, rename, and delete files and folders.
- Develop basic command-line reflexes.

# Introduction to CLI?

**What is a CLI?**

•CLI = Command Line Interface.

•Used to communicate with a computer program, you can input text into a by typing command lines.

•Offers more control, speed, and flexibility than graphical interfaces.

# Introduction to CLI?

## Advantages?

•**Remote access:** CLI applications typically require fewer network resources than GUI ones, appropriate for devices with modest hardware resources or remote server setups.

•**Efficiency:** The CLI lets you perform actions on several files with a single text command, saving you time from having to look for and click on each file individually. You can write scripts that carry out a variety of CLI commands to automate tedious or repetitive tasks.

•**Repetitive operations:** The CLI is great at automating repetitive operations, and you may use a batch file to automate jobs at any time.

•**Power user:** A CLI is suitable for power users since it allows them to execute commands that are not available in the GUI.

# Introduction to CLI?

## Disadvantages?

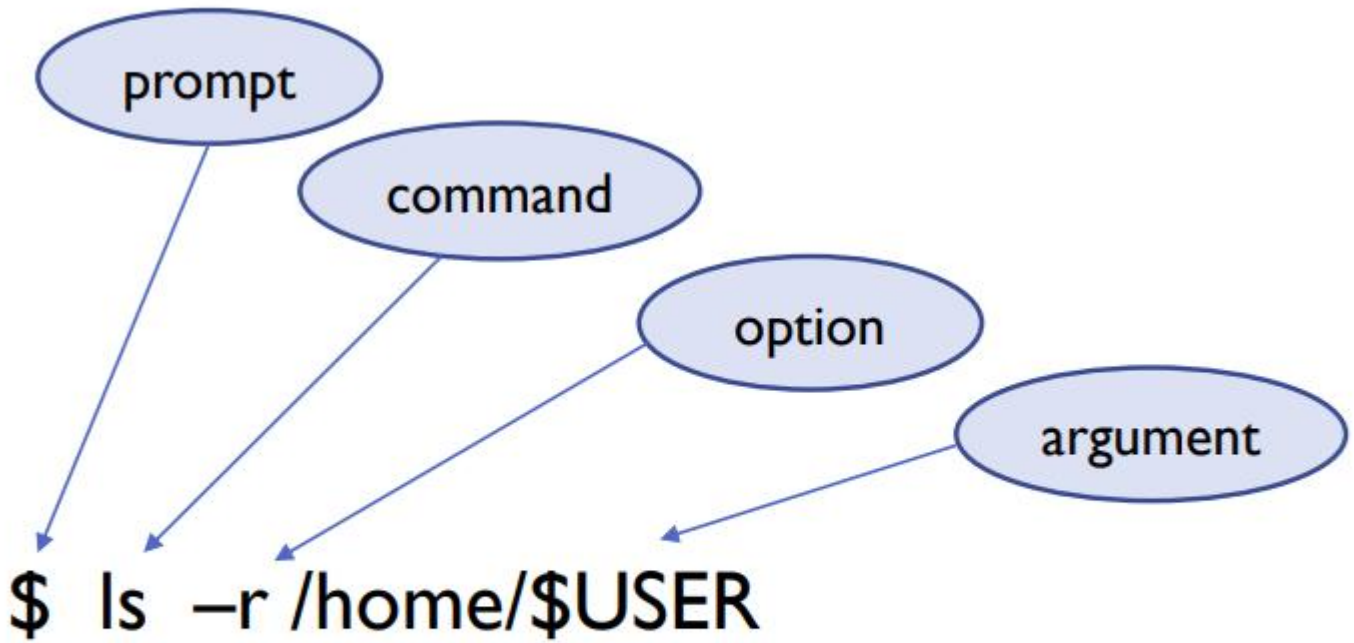
•**Error management:** Errors in command writing might cause files to be deleted or moved to the incorrect location. You run the risk of encountering issues if you unintentionally remove any files, especially if they hold crucial information.

•**Learning curve:** Acquiring command knowledge takes time. The majority of commands are lost if they are not used on a regular basis.

•**Replicate orders:** The majority of commands on the command line interface cannot be undone or reversed. As a result, using commands requires extreme caution. The majority of the time, a command's execution is irreversible and continues.

•**Difficult to remember:** Recalling every command that is used in the CLI might be challenging. Over 100 CLI commands are available in UNIX

# Linux commands



# Linux command basics

- Linux commands are case-sensitive

ls is not the same as LS

- Linux commands may have options that come after the command that start with a “–” and followed by a letter or “- -” and a word:

$ ls –r

$ ls --reverse

- Linux commands may allow for arguments:

$ ls /tmp

- You can run more than one command on the same line by separating the commands with a semicolon (;)

$ ls;date

- man give information about a command

# UNIX commands that give information

**Exercice 1: Type**

- **hostname**
- **date** , can you print only the date? the time?
- **who**
- **whoami**
- **env,** Scroll up and down the screen and make a note of the values of the following Environment variables: HOSTNAME = SHELL= TERM= USER= PATH=
- **uptime**
- **uname** what  does the option r, n ?
- **man uname**.

# UNIX commands that give information

**Exercice 1: Type**

- **hostname** Displays the name of the machine. Each machine on the network is given a unique name by the system administrator.
- **Date** Used to display the current system date and time. date +%D Displays date only date +%T Displays time
- **who** Login details of all users such as their IP, Terminal No, User name.
- **whoiam** Used to display the login details of the user.
- **env** Displays a list of environment variables and their values. We will meet these again later
- **uptime** Displays the time duration that the machine has been turned on.
- **uname** Displays information about the computer system.

Shows version number of the OS (kernel), Displays domain name of the server

- **man uname** The man command gives information on the command that follows it.

# UNIX commands relating to directories

**Exercice 2: Type**

- **pwd** Print (on the screen) the pathname of the working directory (i.e. the directory you are presently in).
- **ls** List contents of directories.
- **cd** Change working directory.
- **mkdir** Make a directory.
- **rmdir** Remove empty directory.
- **mv** Move a directory (also used to rename a directory or file).

**Example of useful options:**

- ls -l : Long format (permissions, size, date).
- ls -a : Show hidden files (those starting with .).
- ls -t : Sort by modification time, newest first.
- ls -r : Reverse the order of the sort.
- ls -lrt : Combined options to show oldest first in detailed view.

# UNI irectories

## Exercice 2: Type

| | | |
|---|---|---|
| 2 | **ls** | List the contents of your current directory (should be empty at this stage). |
| 3 | **mkdir testdir** | Make a subdirectory named 'testdir'. |
| 4 | **cd testdir** | Change directory to the new directory. |
| 5 | **pwd** | Write down the pathname of your current directory. |
| 6 | **ls** | List the contents of your current working directory (should be empty at this stage). |
| 7 | **cd** | The change directory command, cd, on its own will move you back to your 'home directory'. Another way to return to your home directory is to use the command **cd ~** |
| 8 | **pwd** | Write down the pathname of your current directory (it should be your home directory) |
| 9 | **ls** | List the contents of your current directory (this should now show the name of the subdirectory named testdir) |
| 10 | **cd /**<br>**pwd**<br>**ls**<br>**ls -p** | Change to the top most level directory (i.e. the root directory)<br>Confirm where you are using the pwd command.<br>List the contents of the current directory<br>List the contents of the current directory and put a forward slash at the end of any entry that is a directory. |
| 11 | (Task) | Check that the names of the directories under the root directory agree with those listed in the theory page. If not, make a note of any differences. |
| 12 | **cd**<br>**pwd**<br>**ls** | Change to your home directory<br>Confirm where you are using the pwd command.<br>List the contents of the current directory |
| 13 | **cd testdir** | Change to the directory named testdir |
| 14 | **cd ..**<br>**pwd**<br>**ls** | Change to the directory one level up (as specified by the use of two dots).<br>Confirm where you are using the pwd command.<br>List the contents of the current directory |
| 15 | **pwd**<br>**ls**<br>**mkdir testdir/play**<br>**ls**<br>**cd testdir/play**<br>**pwd**<br>**ls**<br>**cd ..**<br>**pwd**<br>**ls**<br>**mv play play2**<br>**ls** | Check you are in your home directory.<br>List the contents of the current directory.<br>Create a directory named **play** underneath the one named testdir.<br>List the contents of the current directory.<br>Change directory into the one just created.<br>Confirm where you are.<br>List the contents of the current directory.<br>Change up one level (ie. back up to the testdir directory)<br>Confirm where you are.<br>List the contents of the current directory.<br>Rename the directory named **play** to **play2.**<br>List the contents of the current directory to check that the renaming has been done. |
| 16 | Note: | The command **cd testdir/play** uses the name of the directory relative to the current directory.<br>To be more precise you could have specified the command as: **cd ./testdir/play** where the single dot means the current directory. |
| 17 | **cd**<br>**pwd**<br>**ls** | Move back up to your home directory.<br>Confirm where you are.<br>List the contents of the current working directory. |
| 18 | **rmdir testdir** | Remove directory. You will find that this command will not work since the directory contains subdirectories. Overcome this by first removing the lower level directories.<br>Later you will be shown a potentially dangerous command that will remove any directory and anything underneath it. |
| 19 | **pwd**<br>**ls** | Check where you are.<br>List the contents of the current directory. |
| 20 | **cd**<br>**mkdir testdir**<br>**mkdir testdir/play** | Recreate the directories again. |

# UNIX commands relating to files

**Commands**

- **cp** Copy file
- **mv** Move a file (also used to rename a file).
- **rm** Remove a file. (Can also be used with the appropriate option to removed a directory and all its contents).
- **touch** create or update a file.

**Example of useful options:**

- rm -i : Prompt before every removal
- rm -r : Recursive delete (for folders)
- rm -f : Force deletion (no confirmation)

# Important shortcuts

**Shortcuts**

- cd .. : Go up one directory
- cd - : Go to the previous directory
- cd ~ : Go to home
- **!!** : repeat last command

# Pattern Matching in the Shell

| Pattern | Meaning |
| --- | --- |
| * | Any number of characters (even none) |
| ? | Exactly one character |
| [wtz] | One character: w, t, or z |
| [a-p] | One character in the range a to p |

🧪 Examples:

```
ls *.txt       # All .txt files
ls ?.sh        # Any 2-character .sh file (e.g., a.sh)
ls [ab]*       # Files starting with a or b
ls file[1-3].txt # file1.txt, file2.txt, file3.txt
```

# LABs

**1.Create working directory “lab” in your home:**
**2. Move to this directory**
**3.Create file1.txt file2.txt file3.txt notes.md data.csv**
**4.Copy all the text file in a “backup” folder**
**5.Rename file3.txt to final.txt**
**6. Move notes.md to a new folder docs/**
**7.Delete docs/ and data.csv**

**Bonus (if time):**

- Try using rm -i to prompt before deleting.
- Try cp -v, mv -i, or rm -v to observe output.
- Use ls *.txt or ls file* to list matching files.
- Use cp *.txt newlocation/ to copy multiple matching files.
- Delete all .txt files at once with rm *.txt (carefully).

# LABs

```
mkdir ~/lab
cd ~/lab
touch file1.txt file2.txt file3.txt notes.md data.csv
mkdir backup
cp *.txt backup/
mv file3.txt final.txt
mkdir docs
mv notes.md docs/
rm data.csv
rm -r docs
```

# More LABs

1.Create a file test.txt
2. Delete it
3.Try to recover it, can you?
4.What option could you use when you remove a file?
5.Create test1.txt test2.txt test1.csv report.md
6.See the difference between ls test?.txt, ls *.md, ls test*.*
7.Using “man cp”, find the option to copy a folder
8. Create a folder “lab”
9. Move all the files into “lab”
10. copy “lab” into “lab_final”

# More LABs

```
# Step 1: Create and delete a file
touch test.txt
rm test.txt

# Step 2: Try recovery
# → Not possible by default. rm deletes permanently.
# To avoid this, use: rm -i (interactive mode) or move to Trash

# Step 3: Create multiple files
touch test1.txt test2.txt test1.csv report.md

# Step 4: Test patterns
ls test?.txt    # Matches test1.txt, test2.txt
ls *.md         # Matches report.md
ls test*.*      # Matches test1.txt, test2.txt, test1.csv

# Step 5: Look up cp with:
man cp          # -r option is used to copy directories

# Step 6: Move and copy folders
mkdir lab
mv test1.txt test2.txt test1.csv report.md lab/
cp -r lab lab_final
```