Other User Interfaces, Commands Chaining, File Redirecting and Shell Scripts

🔷 Other User Interfaces

🔶 Creating Files

- touch filename – Creates an empty file or updates the timestamp if the file exists.
- cat > filename – Creates a new file and allows you to type content (press Ctrl+D to save).
- echo "text" > filename – Creates a file and writes text into it.

🔶 Viewing Files

- cat filename – Displays the content of a file.
- less filename – Opens a file for viewing, allowing scrolling (q to exit).
- more filename – Similar to less, but less flexible.
- head filename – Shows the first 10 lines of a file.
- tail filename – Shows the last 10 lines of a file.
- tail -f filename – Monitors a file in real-time (useful for logs).

🔶 Editing Files

- nano filename – Opens the file in the Nano text editor.
- vim filename – Opens the file in Vim editor.
- echo "new content" >> filename – Appends content to a file.

🔶 Copying, Moving, and Renaming Files

- cp source_file destination_file – Copies a file.
- mv old_name new_name – Renames a file.
- mv filename /path/to/destination/ – Moves a file to another directory.

🔶 Deleting Files

- rm filename – Deletes a file.
- rm -i filename – Prompts before deleting.
- rm -f filename – Deletes without confirmation.

🔶 File Information

- ls -l filename – Shows detailed information about a file.
- stat filename – Displays detailed file statistics.
- file filename – Identifies the file type.
- wc -l filename – Counts the number of lines in a file.

🔶 File Permissions

- chmod 644 filename – Changes file permissions.
- chown user:group filename – Changes file owner.

🔷Commands chaining

🔶 Semicolon (;)

Executes multiple commands **sequentially**, regardless of whether the previous command succeeds or fails.

```
command1 ; command2 ; command3
```

```
echo "Hello" ; ls ; pwd
```

🔶 Logical AND (&&)

Executes the second command **only if** the first command **succeeds** (exit status 0).

```
command1 && command2
```

```
mkdir new_dir && cd new_dir
```

🔶 Logical OR (||)

Executes the second command **only if** the first command **fails** (exit status non-zero).

```
command1 || command2
```

```
ls file.txt || echo "File not found"
```

🔶 AND-OR Combination (&& and ||)

A combination of **AND** and **OR** can simulate an **if-else** condition.

```
command1 && command2 || command3
```

```
mkdir test && echo "Directory created" || echo "Failed to create directory"
```

🔶 Background Execution (&)

```
Runs a command in the background.

command1 &

sleep 10 &

This runs sleep 10 in the background, allowing the shell to continue
accepting commands.
```

🔶 Pipe (|)

```
Passes the output of one command as input to another.

command1 | command2

ls -l | grep ".sh"

This lists files and filters only .sh files.
```

🔷 File Redirecting in Linux

File redirection in Linux allows you to control input and output streams, directing them to files or other commands. This is useful for saving command outputs, processing data, and handling errors.

🔶 Redirecting Standard Output (>, >>)

```
“>” Redirects output to a file (overwrites existing content).
“>>” Redirects output to a file (appends content).
```

```
Examples:

ls > file.txt   # Saves the output of ls to file.txt (overwrites file)

echo "Hello" >> file.txt  # Appends "Hello" to file.txt
```

🔶 Redirecting Standard Input

```
"<" Takes input from a file instead of the keyboard.
```

```
sort < names.txt  # Sorts the content of names.txt and displays output
```