❖ How do we interact with an Operating System?

⇨ We mainly interact in two ways:

1. GUI (Graphical User Interface)

- Uses graphical elements like icons, buttons, menus.
- We perform actions by clicking, dragging, tapping.
- Easy and intuitive for beginners.

Example: Windows desktop, macOS, GNOME, etc.

2. CLI (Command Line Interface)

- Text-based interface
- We type commands instead of clicking
- Preferred by developers/system admins
- Faster for automation, scripting, troubleshooting

Example: Linux terminal, Windows cmd, PowerShell

❖ Terminal vs Shell vs Console (Most Important & Confusing Question)

Terminal (Role: The Interface):

- A program/window that lets you interact with the shell
- Provides the interface to type commands

A program that provides a text-based input/output window. It emulates the old hardware terminals. *Examples:* GNOME Terminal, Konsole, xterm

Shell (Role: The interpreter)**:**

The command-line interpreter that executes commands**.** It's the program that the terminal runs. It takes your commands, processes them (expands variables, handles logic), and calls the OS. Examples: `bash`, `zsh`, `PowerShell`, `fish`.

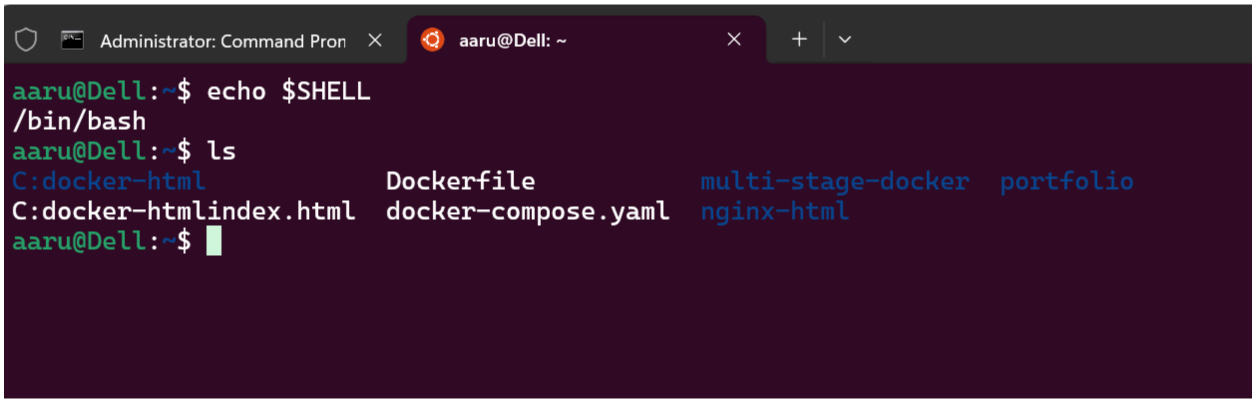
Console (Role: The Physical/Lowest-Level Interface):

The physical or supreme text interface. Originally the physical keyboard & monitor attached directly to a computer. Now also refers to virtual, full-screen text interfaces (e.g., Linux `tty`). It's the *lowest-level* user text interface for the OS.

Simple Flow of Interaction:

- ⇨ You open a **Terminal** (application).
- ⇨ Terminal starts your **Shell** (sh) inside it. (Shell loads when Terminal is opened).
- ⇨ You type a command (ls) into the Terminal window.
- ⇨ The Terminal passes that input to the **Shell**.
- ⇨ The **Shell** interprets the command, executes it (by talking to the OS kernel), and returns the output (e.g., a list of files).
- ⇨ The **Shell** passes that output back to the **Terminal**.
- ⇨ The **Terminal** displays the output in its window.

Summary:

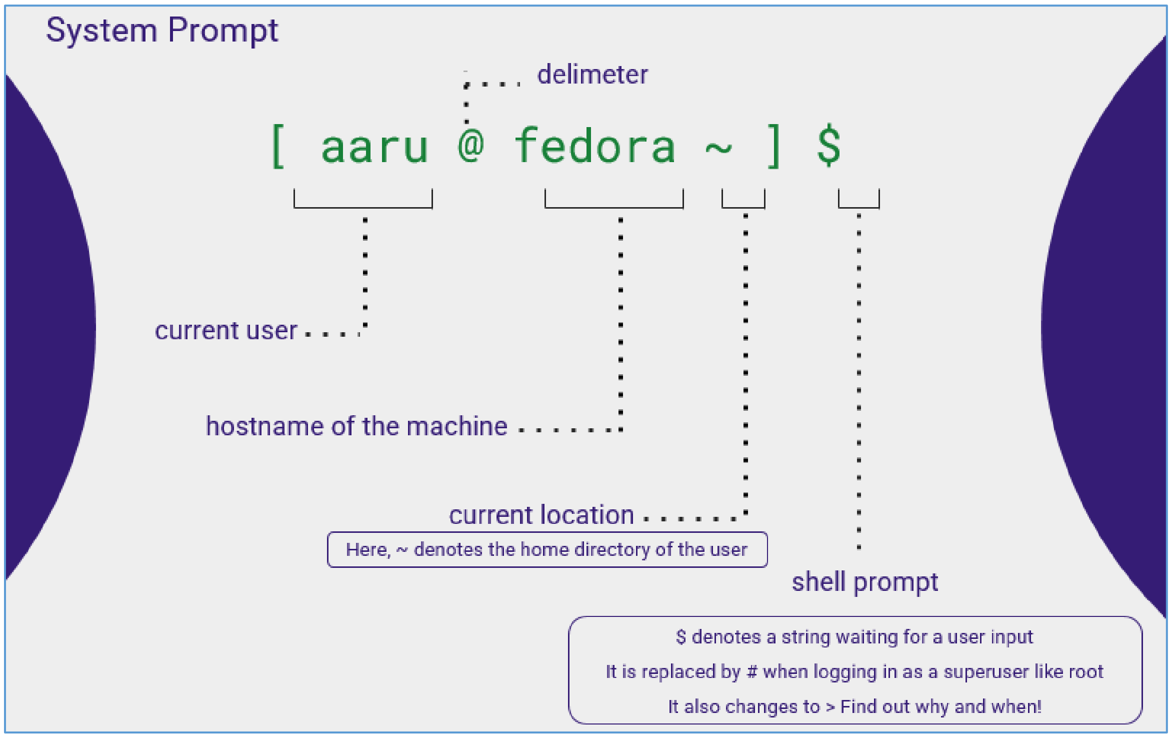


```
aaru@Dell:~$ echo $SHELL
/bin/bash
aaru@Dell:~$ ls
C:docker-html             Dockerfile            multi-stage-docker  portfolio
C:docker-htmlindex.html  docker-compose.yaml   nginx-html
aaru@Dell:~$
```

Terminal is the **entire graphical window** itself, which allows us to interact with the shell. It's the program that simulates a physical terminal. Shell is the **program** that interprets the commands we type. Here I typed ls, it takes input, and gives output. System prompt is the text displayed by the shell **before** we type a command, indicating that the shell is ready to accept input. In the image, the system prompt is **aaru@Dell:~$.**

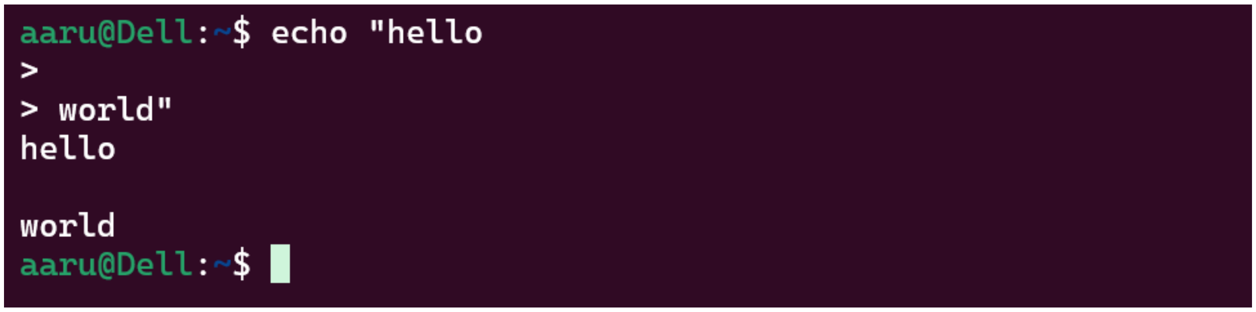
System Prompt?

A system prompt is the line displayed by the shell in a terminal that indicates the system is ready to accept user commands. It usually shows information like username, hostname, current directory, and a symbol indicating user type ($ for normal users, # for root).



System Prompt
delimeter
[aaru @ fedora ~] $
current user
hostname of the machine
current location
Here, ~ denotes the home directory of the user
shell prompt
$ denotes a string waiting for a user input
It is replaced by # when logging in as a superuser like root
It also changes to > Find out why and when!

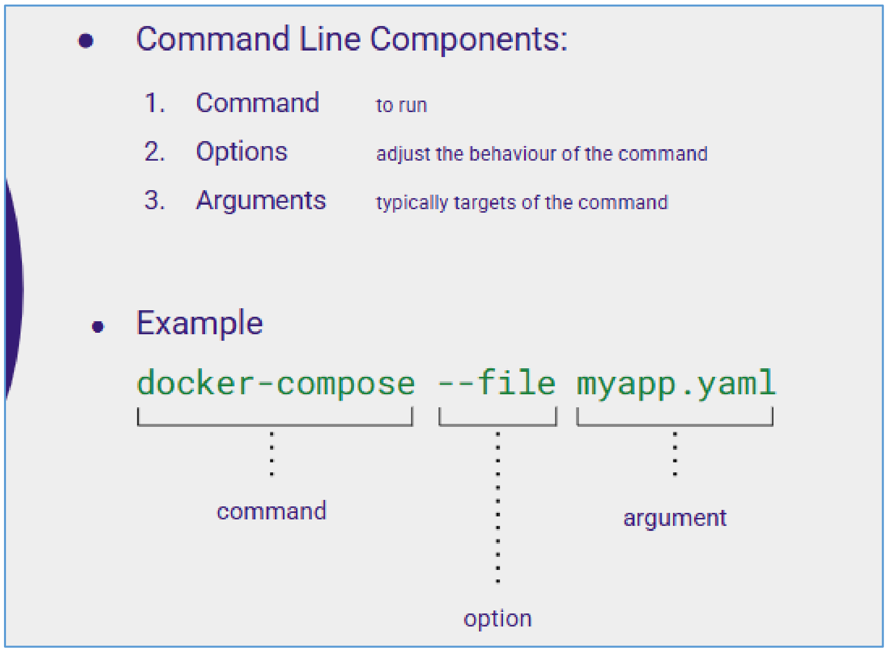
The **“>”** prompt appears automatically when the shell detects your command isn’t finished and waits for you to continue writing (**Continuation prompt**).



```
aaru@Dell:~$ echo "hello
>
> world"
hello

world
aaru@Dell:~$ 
```

Command Line Components:



- Command Line Components:
  1. Command — to run
  2. Options — adjust the behaviour of the command
  3. Arguments — typically targets of the command

- Example

```
docker-compose --file myapp.yaml
```

Note: Argument can be of both command and options

Example: grep -i error logfile.txt   Where,

grep = command
-i = option
error = argument to option
logfile.txt = argument to command