What is Shell?

A brief introduction to Shell and its types.

> A shell is a macro processor or a command language interpreter that primarily translates the commands, written by the user in the terminal, into system actions that are executed, which can also automatically run in programs called shell scripts.

A shell is not an operating system. It is a way to interface with the operating system and run commands.

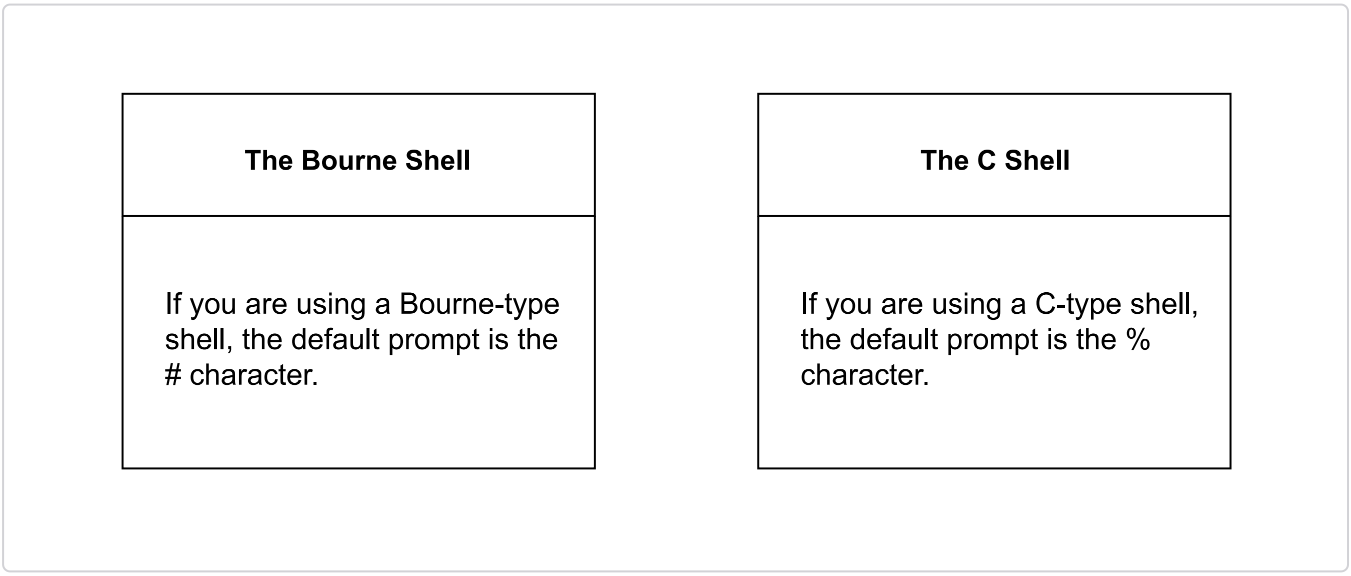
The terminal window in our computer contains a shell that allows you to process information, store or retrieve data and much more by interacting with a computer via entering commands. For example,

- Retrieving a list of files or directories
- Looking up today's date and time
- Getting current directory location
- Making, copying or deleting files
- Sorting files

And much more complicated tasks!

Types of Shell

In UNIX, there are two major types of shell:


The Bourne Shell
If you are using a Bourne-type shell, the default prompt is the # character.
The C Shell
If you are using a C-type shell, the default prompt is the % character.

Why Shell Scripting?

Just as you can type and execute commands on a Shell terminal, you can put a set of commands in a plain text file with the extension of `.sh`. These files are shell scripts which can be executed like normal `GNU` or `Unix`

- A Unix shell is both, a command interpreter and a programming language. As a *command interpreter*, it provides the user interface to the substantial set of GNU utilities. The *programming language features* allow these utilities to be combined
- Shell scripting allows users to automate their tasks. Script files containing commands can be executed as commands themselves. These customized commands have the same status as system commands in directories like `/bin`. Hence, the users can create their own customized environments to carry out tasks
- Shell scripts can run in an interactive and non-interactive mode
- A shell allows synchronous and asynchronous execution of `GNU` commands
- Shell scripting allows rapid prototyping
- Shell scripting reduces extensive, complex and repetitive sequences of command into a simple command

Typical Examples of Shell Programming

- Allows to create user accounts

- Writing application startup scripts, especially unattended applications
- Find out what processes are eating up your system resources
- Find out available and free memory
- Find out all logged in users and what they are doing
- Writing system boot scripts
- User administration as per your security policies
- Find out information about local or remote servers
- Creating application package installation tools
- Automation of customized processes