GETTING STARTED

WITH

SHELL SCRIPTING

➢ PROGRAMMING LANGUAGE:

- A **programming language** is a formal computer **language** or constructed language designed to communicate instructions to a machine, particularly a computer.
- **Programming languages** can be used to create programs to control the behaviour of a machine or to express algorithms.

➢ SCRIPTING LANGUAGE:

- A script or **scripting language** is a computer **language** with a series of commands within a file that is capable of being executed without being compiled.
- Good examples of server-side **scripting languages** are Shell, Perl, PHP, and Python.
- The best example of a client-side **scripting language** is JavaScript.

➢ THE SHELL:

- "Shell" is the UNIX term for a user interface to the system—something that lets you communicate with the computer via the keyboard and the display.
- The shell's job, then, is to translate the users command line into operating system instructions. Remember that the shell itself is not UNIX-Just the user interface to it.
- Unix is one of the first operating system(O/S) to make the user interface independent of the operating system.
- The major shell was the Bourne shell; it was included in the first popular version of UNIX, version 7, Starting in 1979.

➢ TYPES OF SHELLS:

SHELL NAME	DEVELOPED BY	PROMPT	INTERPRETER NAME
Bourne Shell	Stephen Bourne	$	sh
Korn Shell	David Korn	$	ksh
C shell	Bill Joy	%	csh
Bash Shell	Stephen Bourne	$	bash
Z Shell	Paul	$	zsh

→ To Know the shell version: $**echo $BASH_VERSION**.

→ **To see the parent shell of current user:** $echo $SHELL

→ **To List available shells:** $ls /bin/*.sh (or) $cat /etc/shells

→ **To shift to Bourne shell**: $sh

→ **To verify current shell:** $0

➢ BASH SHELL FEATURES:

- Job control, including the **fg** and **bg** commands and the ability to stop jobs with **CTRL−Z.**
- Brace expansion, for generating arbitrary strings. Tilde expansion, a shorthand way to refer to directories.
- Aliases, which allow you to define shorthand names for commands or command lines.
- Command history, which lets you recall previously entered commands.
- Command−line editing, allowing you to use vi− or emacs−style editing commands on your command lines.

➢ SHELL PROGRAMMING:

- A shell programming is nothing but a series of such commands. (or) When a group of commands have to be executed regularly, they should be stored in a file, and the file executed as a shell program. Though it's not mandatory, using the .sh extension for shell scripts makes it easy to much them with wild cards.
- A shell script needs to have execute permission when invoked by its name. It's not compiled to a separate executable file as a 'C' program is. It runs in interpretive mode and in a separate child process. The calling process forks a sub shell, which reads the script file and loads each statement into memory when it is to be executed.
- Shell scripts are thus slower than compiled programs, but speed is not a constraint with certain jobs.

➢ THE MAIN CONCEPT OF SHELL SCRIPTING IS:

- To handle text files i.e. it has given very rich text processing features and text utilities.
- It saves lot of work time.
- It creates new Unix/Linux commands.

➢ WHEN TO USE SHELL SCRIPTS:

- Customizing your work environment.
- Automating your daily tasks.
- Automating repetitive tasks.
- Executing important system procedures like shutdown, formatting a disk, creating a file system, mounting a file system.
- Performing same operation on many files.

➢ CREATING A SHELL SCRIPT:

STEP1: CREATE A FILE USING A TEXT EDITOR LIKE **VI / VIM.**

STEP2: NAME SCRIPT FILE WITH EXTENSION **".sh".**

STEP3: START THE SCRIPT WITH **"#!/bin/sh".**

STEP4: WRITE SOME CODE.

STEP5: SAVE THE SCRIPT FILE AS **"filename.sh".**

STEP6: SET EXECUTE PERMISSIONS WITH **"chmod a+x filename.sh".**

STEP7: EXECUTING THE SCRIPT TYPE **"bash filename.sh"** (or) **"./filename.sh**".

EXAMPLE: Create a file with name "script.sh"

```
$vim script.sh
  #!/bin/sh
  clear
  date
  whoami
```

HOW TO EXECUTE SHELL SCRIPT:

To set execute permissions for script: $**chmod a+x script.sh**

To run a shell script: $**sh script.sh** (or) $**./script.sh**

SHE-BANG LINE (#!/BIN/SH):

- The first line of script.sh contains a string beginning with #!. This is not a comment line. It is called interpreter line, hash-bang (or) she-bang line.
- When the script executes, the login shell reads this line first to determine pathname of the program to be used for running a script. Here, the login shell spawns a Bourne shell which actually executes each statement in **sequence.**
- If you don't provide the she-bang line, the login shell will spawn a child of its own type to run the script-which may not be the shell you want.

❖ ECHO COMMAND:

- It is a shell keyword. It is used for to write data to the output screen.

```
$echo “Scripting is fun!”
$echo “Today date is: date”
$echo Today date is: `date` (or) $date
```

NOTE: Here `date` or $date is the command substitution operator.

```
$echo -e "Hello \n Linux" ("\n" is a new line)
```

➢ INTERACTIVE SHELL SCRIPTS:

- The **"read"** statement is the shell's internal tool for taking input from the user, i.e making scripts interactive.
- It is used with one or more variables that are assigned by keyboard input.

```
#!/bin/sh
echo What is your Name \?
read name
echo Hello $name. Happy Programming.
```

NOTE: The question mark must be preceded by backslash to convey to the shell that here the '?' is to be treated as an ordinary character. If not preceded by a '\', for the shell '?' stands for all files in the current directory whose names are one character long.