# Shell Programming

A Unix shell is a command-line interpreter or shell that provides a command-line user interface for Unix-like operating systems. It is an interface between the user and the operating system itself. The shell is used by the operating system to control the execution of the system using shell scripts.

Types of Shells: Different people implemented the interpreter function of the shell in different ways. Therefore, there exist various types of shells. For example, Bourne shell, C shell, Korn shell. These three most widely used Unix shells support processes (both foreground and background), pipes, filters, directories, and other similar standard features of Unix.

Bash is a command language interpreter. It is widely available on various operating systems, and is a default command interpreter on most Linux systems. The name is an acronym for the "**B**ourne-**A**gain **Sh**ell ".

Scripting allows for an automatic commands execution that would otherwise be executed interactively one-by-one.

We have seen that the Unix commands are executed when they are typed in at the shell prompt. A shell program is nothing but a series of such commands. The shell programming language incorporates most of the features that most modern day programming languages offer. For example, it has local and global variables, control instructions, functions, etc. So if you already know a programming language you would find many familiar concepts in shell programming. If we are to execute a shell program we don't need a separate compiler. The shell itself interprets the commands in the shell program and executes them.

*Example* 1: Create a file **test1.sh** which contains the commands **ls**, **who**, **pwd** in each line. Run this file at the shell prompt.

```
$ bash test1.sh
```

*Example* 2: Create a file **test2.sh** that contains the following statements:

```
echo -e “Please enter your name”
read name
echo “Nice to meet you $name”
```

Run this file at the shell prompt.

```
$ bash test2.sh
```

*Note: echo, read are shell keywords* (*or Reserved words*). *name (in the second line and third line) is a shell variable.*

*Example 3: You may also Run at the shell prompt.*

```
$ read name
$ echo $name
```

*Example 4: You may also initialize a name to the shell variable* ***name****, and also display that name.*

```
$ name=”VNIT Nagpur”
$ echo $name
```

*Example 5: You initialize a value to a shell variable* ***a****, and display the value of the variable.*

```
$ a=200
$ echo $a

$ echo a
```

*Note: Omitting the $ before the shell variable* ***a*** *would simply treat* ***a*** *as a character to be echoed.*

*Exercise 1: Write a shell script which reads <your name> in a shell variable* ***name****, <your address> in three lines in three shell variables namely* ***address_line1****,* ***address_line2****,* ***address_line3****; Display those at the shell prompt.*

*Exercise 2: Write a shell script which reads two integer values for two shell variables say,* ***a****,* ***b****, and perform the following operations on these two variables:* ***addition, subtraction, multiplication, division, and modulo division.***

*Exercise 3:  The length & breadth of a rectangle and radius of a circle are input through the keyboard. Write a shell script to calculate the area and perimeter of the rectangle, and the area and circumference of the circle.*

*Exercise 4: The length of the three sides of a triangle is input through the keyboard. Write a shell script to calculate the area of the triangle.*

Exercise 5: Write a shell script which receives two filenames as arguments. It should check whether the two file's contents are same or not. If they are the same then the second file should be deleted.

Exercise 6: Write a shell script which will receive any number of filenames as arguments. The shell script should check whether every argument supplied is a file or directory. If it is directory it should be appropriately reported. If it is a file name then name of the file as well as the number of lines present in it should be reported.