Unix / Linux - What is Shells?

A **Shell** provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

Shell Prompt

The prompt, **$**, which is called the **command prompt**, is issued by the shell. While the prompt is displayed, you can type a command.

Shell reads your input after you press **Enter**. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

Following is a simple example of the **date** command, which displays the current date and time −

```
$date
Thu Jun 25 08:30:19 MST 2009
```

You can customize your command prompt using the environment variable PS1 explained in the Environment tutorial.

Shell Types

In Unix, there are two major types of shells −

- **Bourne shell** − If you are using a Bourne-type shell, the **$** character is the default prompt.
- **C shell** − If you are using a C-type shell, the % character is the default prompt.

The Bourne Shell has the following subcategories −

- Bourne shell (sh)
- Korn shell (ksh)
- Bourne Again shell (bash)
- POSIX shell (sh)

The different C-type shells follow −

- C shell (csh)

- TENEX/TOPS C shell (tcsh)

The original Unix shell was written in the mid-1970s by Stephen R. Bourne while he was at the AT&T Bell Labs in New Jersey.

Bourne shell was the first shell to appear on Unix systems, thus it is referred to as "the shell".

Bourne shell is usually installed as **/bin/sh** on most versions of Unix. For this reason, it is the shell of choice for writing scripts that can be used on different versions of Unix.

In this chapter, we are going to cover most of the Shell concepts that are based on the Borne Shell.

Shell Scripts

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by **#** sign, describing the steps.

There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include functions.

We are going to write many scripts in the next sections. It would be a simple text file in which we would put all our commands and several other required constructs that tell the shell environment what to do and when to do it.

Shell scripts and functions are both interpreted. This means they are not compiled.

Example Script

Assume we create a **test.sh** script. Note all the scripts would have the **.sh** extension. Before you add anything else to your script, you need to alert the system that a shell script is being started. This is done using the **shebang** construct. For example −

```
#!/bin/sh
```

This tells the system that the commands that follow are to be executed by the Bourne shell. *It's called a shebang because the* **#** *symbol is called a hash, and the ! symbol is called a bang*.

To create a script containing these commands, you put the shebang line first and then add the commands −

```
#!/bin/bash
pwd
ls
```

Shell Comments

You can put your comments in your script as follows −

```
#!/bin/bash

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:
pwd
ls
```

Save the above content and make the script executable −

```
$chmod +x test.sh
```

The shell script is now ready to be executed −

```
$./test.sh
```

Upon execution, you will receive the following result −

```
/home/amrood
index.htm  unix-basic_utilities.htm  unix-directories.htm
test.sh    unix-communication.htm    unix-environment.htm
```

Note − To execute a program available in the current directory, use **./program_name**

Extended Shell Scripts

Shell scripts have several required constructs that tell the shell environment what to do and when to do it. Of course, most scripts are more complex than the above one.

The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.

The following script uses the **read** command which takes the input from the keyboard and assigns it as the value of the variable PERSON and finally prints it on STDOUT.

```
#!/bin/sh

# Author : Zara Ali
# Copyright (c) Tutorialspoint.com
# Script follows here:

echo "What is your name?"
```

```
read PERSON
echo "Hello, $PERSON"
```

Here is a sample run of the script −

```
$./test.sh
What is your name?
Zara Ali
Hello, Zara Ali
$
```