UNIX / LINUX - ENVIRONMENT

In this chapter, we will discuss in detail about the Unix environment. An important Unix concept is the **environment**, which is defined by environment variables. Some are set by the system, others by you, yet others by the shell, or any program that loads another program.

A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

For example, first we set a variable TEST and then we access its value using the **echo** command –

```
$TEST="Unix Programming"
$echo $TEST
```

It produces the following result.

```
Unix Programming
```

Note that the environment variables are set without using the **$** sign but while accessing them we use the $ sign as prefix. These variables retain their values until we come out of the shell.

When you log in to the system, the shell undergoes a phase called **initialization** to set up the environment. This is usually a two-step process that involves the shell reading the following files –

- /etc/profile
- profile

The process is as follows –

- The shell checks to see whether the file **/etc/profile** exists.
- If it exists, the shell reads it. Otherwise, this file is skipped. No error message is displayed.
- The shell checks to see whether the file **.profile** exists in your home directory. Your home directory is the directory that you start out in after you log in.
- If it exists, the shell reads it; otherwise, the shell skips it. No error message is displayed.

As soon as both of these files have been read, the shell displays a prompt –

```
$
```

This is the prompt where you can enter commands in order to have them executed.

Note – The shell initialization process detailed here applies to all **Bourne** type shells, but some additional files are used by **bash** and **ksh**.

The .profile File

The file **/etc/profile** is maintained by the system administrator of your Unix machine and contains shell initialization information required by all users on a system.

The file **.profile** is under your control. You can add as much shell customization information as you want to this file. The minimum set of information that you need to configure includes –

- The type of terminal you are using.

- A list of directories in which to locate the commands.
- A list of variables affecting the look and feel of your terminal.

You can check your **.profile** available in your home directory. Open it using the vi editor and check all the variables set for your environment.

Setting the Terminal Type

Usually, the type of terminal you are using is automatically configured by either the **login** or **getty** programs. Sometimes, the auto configuration process guesses your terminal incorrectly.

If your terminal is set incorrectly, the output of the commands might look strange, or you might not be able to interact with the shell properly.

To make sure that this is not the case, most users set their terminal to the lowest common denominator in the following way –

```
$TERM=vt100
$
```

Setting the PATH

When you type any command on the command prompt, the shell has to locate the command before it can be executed.

The PATH variable specifies the locations in which the shell should look for commands. Usually the Path variable is set as follows –

```
$PATH=/bin:/usr/bin
$
```

Here, each of the individual entries separated by the colon character **(:)** are directories. If you request the shell to execute a command and it cannot find it in any of the directories given in the PATH variable, a message similar to the following appears –

```
$hello
hello: not found
$
```

There are variables like PS1 and PS2 which are discussed in the next section.

PS1 and PS2 Variables

The characters that the shell displays as your command prompt are stored in the variable PS1. You can change this variable to be anything you want. As soon as you change it, it'll be used by the shell from that point on.

For example, if you issued the command –

```
$PS1='=>'
=>
=>
=>
```

Your prompt will become =>. To set the value of **PS1** so that it shows the working directory, issue the command –

```
=>PS1="[\u@\h \w]\$"
[root@ip-72-167-112-17 /var/www/tutorialspoint/unix]$
[root@ip-72-167-112-17 /var/www/tutorialspoint/unix]$
```

The result of this command is that the prompt displays the user's username, the machine's name (hostname), and the working directory.

There are quite a few **escape sequences** that can be used as value arguments for PS1; try to limit yourself to the most critical so that the prompt does not overwhelm you with information.

Sr.No.	Escape Sequence & Description
1	**\t** Current time, expressed as HH:MM:SS
2	**\d** Current date, expressed as Weekday Month Date
3	**\n** Newline
4	**\s** Current shell environment
5	**\W** Working directory
6	**\w** Full path of the working directory
7	**\u** Current user's username
8	**\h** Hostname of the current machine
9	**\#** Command number of the current command. Increases when a new command is entered
10	**\$** If the effective UID is 0 (that is, if you are logged in as root), end the prompt with the # character; otherwise, use the $ sign

You can make the change yourself every time you log in, or you can have the change made automatically in PS1 by adding it to your **.profile** file.

When you issue a command that is incomplete, the shell will display a secondary prompt and wait for you to complete the command and hit **Enter** again.

The default secondary prompt is **>** (the greater than sign), but can be changed by re-defining the **PS2** shell variable –

Following is the example which uses the default secondary prompt –

```
$ echo "this is a
> test"
this is a
test
$
```

The example given below re-defines PS2 with a customized prompt –

```
$ PS2="secondary prompt->"
$ echo "this is a
secondary prompt->test"
this is a
test
$
```

Environment Variables

Following is the partial list of important environment variables. These variables are set and accessed as mentioned below –

Sr.No.	Variable & Description
1	**DISPLAY** Contains the identifier for the display that **X11** programs should use by default.
2	**HOME** Indicates the home directory of the current user: the default argument for the cd **built-in** command.
3	**IFS** Indicates the **Internal Field Separator** that is used by the parser for word splitting after expansion.
4	**LANG** LANG expands to the default system locale; LC_ALL can be used to override this. For example, if its value is **pt_BR**, then the language is set to (Brazilian) Portuguese and the locale to Brazil.
5	**LD_LIBRARY_PATH**

	A Unix system with a dynamic linker, contains a colonseparated list of directories that the dynamic linker should search for shared objects when building a process image after exec, before searching in any other directories.
6	**PATH** Indicates the search path for commands. It is a colon-separated list of directories in which the shell looks for commands.
7	**PWD** Indicates the current working directory as set by the cd command.
8	**RANDOM** Generates a random integer between 0 and 32,767 each time it is referenced.
9	**SHLVL** Increments by one each time an instance of bash is started. This variable is useful for determining whether the built-in exit command ends the current session.
10	**TERM** Refers to the display type.
11	**TZ** Refers to Time zone. It can take values like GMT, AST, etc.
12	**UID** Expands to the numeric user ID of the current user, initialized at the shell startup.

Following is the sample example showing few environment variables –

```
$ echo $HOME
/root
]$ echo $DISPLAY

$ echo $TERM
xterm
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/home/amrood/bin:/usr/local/bin
$
```