Understand basic steps to write a shell scripts

Following steps are required to write shell script:

1. Create a script
2. Setup a executable permission on a script
3. Run a script
4. Debug a script if required

Let us try to understand each step in depth, if you are already familiar with above four steps please skip to next section.

Create a script

As discussed earlier shell scripts stored in plain text file, generally one command per line. You can use text editor such as vi or emacs. I recommend using vim (Vi Improved) as it is equipped with features such as syntax highlighting and indenting. Most modern Linux (or *BSD) distribution comes with vim. You can start vi or vim from shell prompt by typing any one of the following command:

```
$ vi  myscript.bash
$ vi  myscript.sh
$ vim  myscript.bash
```

Make sure you use .bash or .sh file extension for each script. This ensures easy identification of shell script.

Setup executable permission

Once script is created, you need to setup executable permission on a script. Why to setup executable permission, might be next question in your mind, right?

Simple,

- Without executable permission, running a script is almost impossible
- Besides executable permission, script must have a read permission

Syntax to setup executable permission:

```
chmod permission your-script-name
```

Examples:

```
$ chmod +x your-script-name
$ chmod 755 your-script-name
```

Run a script (execute a script)

Now your script is ready with proper executable permission on it. Next, test script by running it.

Syntax:

```
bash your-script-name
sh your-script-name
./your-script-name
```

Examples:

```
$ bash bar
$ sh bar
$ ./bar
```

In last example, you are using **.** (dot) command which read and execute commands from filename in the current shell. If filename does not contain a slash, file names in PATH are used to find the directory containing filename. For example:

```
./bar
```

bar script executed from current directory. The specialty of dot (.) command is you do not have to setup an executable permission on script.

Debug a script if required

While programming shell sometimes you need to find out errors (bugs) in shell script and correct all errors (remove errors i.e. debug script). For this purpose you can pass **-v** and **-x** option to sh/bash command to debug the shell script. General syntax is as follows: Syntax:

```
sh   option   { shell-script-name }
```

OR

```
bash   option   { shell-script-name }
```

Where,

- **-v**: print shell input lines as they are read
- **-x**: after expanding each simple-command, bash displays the expanded value of PS4 system variable, followed by the command and its expanded arguments.

Debugging discussed later in depth. Now you are ready to write first shell script that will print "Knowledge is Power" on screen. See the common vi command list , if you are new to vi.

```
$ vi first
#
# My first shell script
#
clear
echo "Knowledge is Power"
```

After saving the above script, you can run the script as follows:

```
$ ./first
```

This will not run script since we have not set execute permission for our script*first*; to do this type command

```
$ chmod 755 first
$ ./first
```

First screen will be clear, then Knowledge is Power is printed on screen.

Script Command(s)	Meaning
$ vi first	Start vi editor
# # My first shell script #	# followed by any text is considered as comment. Comment gives more information about script, logical explanation about shell script. *Syntax:*

	# comment-text
clear	clear the screen
echo "Knowledge is Power"	To print message or value of variables on screen, we use echo command, general form of echo command is as follows *syntax:* echo "Message"

How Shell Locates the file (My own bin directory to execute script)

Tip: For shell script file try to give file extension such as .sh, which can be easily identified by you as shell script.

Exercise:
1)Write following shell script, save it, execute it and note down the it's output.

```
$ vi ginfo
#
#
# Script to print user information who currently login , current date & time
#
clear
echo "Hello $USER"
echo "Today is \c ";date
echo "Number of user login : \c" ; who | wc -l
echo "Calendar"
cal
exit 0
```