1. Shell Scripting – Introduction & Script Creation

Shell scripting is one of the programming languages like any other programming language which can be implemented using algorithm or flow control statements. Shell script is a collection of shell commands. Based on the closeness to the hardware, different programming languages can be categorized as below:

1. Low level / Assembly language:

   Assembly codes are very close to the hardware. These codes completely depend on the architecture and its instruction set. To write any assembly code the instruction set to be known first. Instruction set of every architecture is different from one another.

   Ex: As a part of product development if the architecture is changed from Intel to ARM, to implement same logic which is implemented using Intel on ARM, instruction set and its working of ARM must be known.

2. High level language:

   High level languages are completely independent of architecture. The same code can be executed on different systems. These codes cannot interact with hardware.

   Ex: Shell script

3. Middle level language:

   These languages can be converted either to interact with a specific architecture (low level codes) or can be written independent of the architecture (high level codes). To implement the same logic to interact with different architectures, the code should be modified as per the specific architecture.

   Ex: C program

The choice of any particular programming language completely depends on the requirement. Every language has its own pros and cons.

For example, speed of execution of assembly codes are comparatively high as these codes are very close to the hardware. But, to learn and implement assembly code is very difficult especially if the architecture has to be changed.

Difference between programs and scripts:

Shell scripts are executable files which require an interpreter to execute. Shell itself is an application which works as the interpreter. Interpreter executes the code line by line.

Programs are needed to be compiled to generate the executable file by the compiler. In this case, one entire file will be executed and it's not line by line execution. Ex. C program.

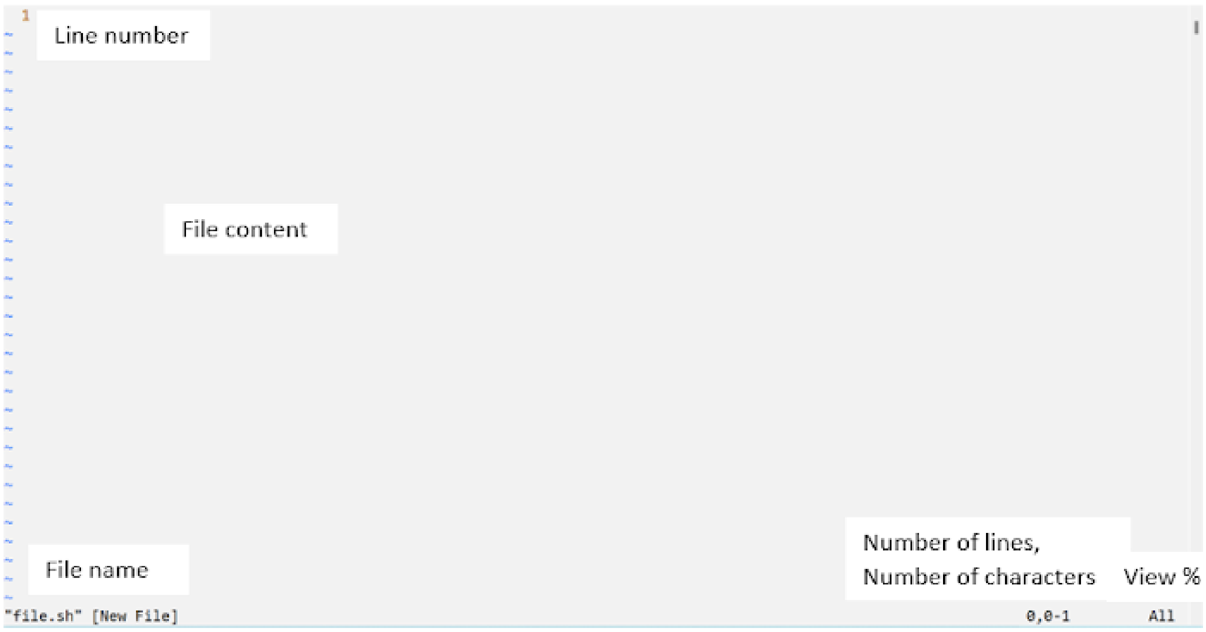
Shell scripts are mainly used for all automated tasks, especially for the automated testing tools in industries.

Creation of shell script:

- All scripts will be done using vim editor
    - If vim is not installed in your system, then install it using the following command line. Execute it on terminal:

sudo apt-get update; sudo apt install vim

- Create shell script using vim: (vim editor should be installed first)
    - vim filename.sh (file extension should be .sh)
    - save file using :w in Esc mode
    - the file won't be created unless and until its saved manually



```
"file.sh" [New File]                                              0,0-1         All
```

- :q in Esc mode can be done to close file
- To add content, edit the file using insert mode; press I key on keyboard.
- After editing file save and close file using :wq in Esc mode

Importance of shebang:

Shebang is an integral part of every script. #! Followed by absolute path of the interpreter is known as the shebang.

Ex: #!absolute path of bash → **#!/bin/bash** is the shebang for a bash script.

If the shebang and file extensions are missing while executing any script then shell will consider that as a shell script by default and that may result in erroneous output as every shell has different features and syntax.

Methods to execute a bash script:

1. Using command bash:

   Usage: bash file.sh

   This method will execute the shell script irrespective of whether the script has executable permission or not.

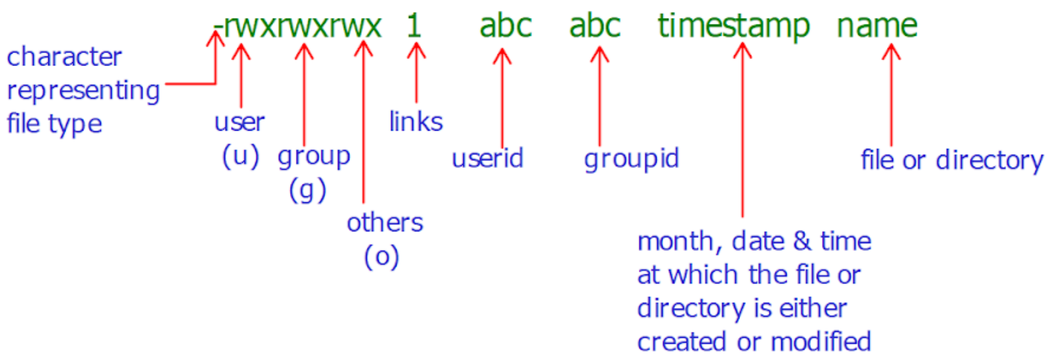
2. Usage: ./filename

   This method can be used for any file which is executable

   If a file is not executable, then the execute permission can be added using chmod command.

chmod:

chmod(change mod bits) command is to modify existing permissions for a file or directory. The read(r), write(w) and execute(x) permissions can be modified either using three-digit numbers in the range 0-7 or using characters u(suer), g(group), o(others) or (a)all where all stands for (u + g + o).


-rwxrwxrwx
1
abc
abc
timestamp
name
character representing file type
user (u)
group (g)
others (o)
links
userid
groupid
month, date & time at which the file or directory is either created or modified
file or directory

An example of long list (ls -l) is given above. Here all three permission sets (rwx) can be either modified using the number or characters. If the permission needs to be modified for any one of user or group or others or for a combination of any two, its possible with the operation using characters.

Ex: chmod ug+wx file.sh

Here the write and execute permissions will be added to the file.sh

chmod -R g-r Dir

Here read permission will be removed for Dir directory and its contents.

```
chmod 653 file.sh
```

Here permission sets will be modified as rw-r-x-wx

Always modifying permission bits using numbers will modify the existing permissions of all (u + g + o).