BASH SCRIPTING

Basics

Contents

- Bash Shell Scripting Overview
- Basics Steps to Write and Execute Bash Shell Scripting With an Example
- List of General Purpose Commands and Help to Understand the Usage of a Command
- Redirection Operators and STDIN, STDOUT & STDERR
- Complete Echo Command
- Working with Variables
- How to Access Variables Which are Defined Another File?
- How to Store the Exit Status of a Command into a Variable?

Introduction

A Bash script is a plain text file which contains a series of commands.

Anything you can run normally on the command line can be put into a script and it will do exactly the same thing. Similarly, anything you can put into a script can also be run normally on the command line and it will do exactly the same thing.

SHEBANG LINE

Each Bash based Linux script starts by the line-

1. #! /bin/bash

Where #! is referred to as the ***shebang*** and rest of the line is the path to the interpreter specifying the location of bash shell in our operating system.

> Under the Unix-like operating systems, when a script with a shebang runs as a program, the program loader parses the rest of the lines with the first line as an interpreter directive. So, SheBang denotes an interpreter to execute the script lines, and it is known as the ***path directive*** for the execution of different kinds of Scripts like Bash, Python, etc.

Basics Steps to Write and Execute Bash Shell Script

How to create a Bash Script?

- To create an empty bash script, first, change the directory in which you want to save your script using **cd** command. Try to use text editor like ***gedit*** in which you want to type the shell commands.
- Use **touch** command to create the zero bytes sized script.
- Add the shebang line

Here, **.sh** is suffixed as an extension that you have to provide for execution.

Run the script.

You can run the script in the following ways:

```
./hello_world.sh

bash hello_world.sh
```

Example-Print the value of a variable

```
#!/usr/bin/env bash

name="John"

echo "Hello $name!"
```

Echo Basics

```
name="John"

echo "${name}"

echo "${name/J/j}"    #=> "john" (substitution)

echo "${name:0:2}"    #=> "Jo" (slicing)

echo "${name::2}"     #=> "Jo" (slicing)

echo "${name::-1}"    #=> "Joh" (slicing)

echo "${name:(-1)}"   #=> "n" (slicing from right)

echo "${name:(-2):1}" #=> "h" (slicing from right)

echo "${food:-Cake}"  #=> $food or "Cake"
```

How to Store the Exit Status of a Command into a Variable

"$?" is a variable that holds the return value of the last executed command.

"echo $?" displays 0 if the last command has been successfully executed and displays a non-zero value if some error has occurred.

The bash sets "$?" To the exit status of the last executed process.

Assignment