**Linux Shell Scripting: A project-based approach to learning**

13 May 2021

15:32

**SIXTH SCRIPT**

1. Positional Parameters
2. Arguments for loops
3. Special Parameters

* #! /bin/bash - Writing this at the top of the script will make it a special directive for shell to know that it should use /bin/bash as the interpreter. Shell will not consider this line at the top to be comment however, after this if any statement begins with # sign, it will be regarded as a comment
* Shell is the GUI/CLI between the user and the operating system and bash (Bourne Again Shell) is the command line interpreter used for GUI linux oeprating system. Bash is shell.
* Echo "You executed this command ${0}”. In this command, ${0} represents the 0th positional parameter/ variable in the command line.
* Parameter vs argument: parameter is a variable used in the shell script and argument is the value passed to the shell script to facilitate its execution.
* So, the argument passed in the command line becomes the parameter's value in the shell script
* Tip!
  + When we type any command in shell. Shell first searches the command in its list of functions, if not, then, it searches the command in list of built-in shell commands. If it does not find the searched/ entered command in either the function list or shell built-commands list, it then finally, searches inside the ${PATH}. If nothing works out then, shell displays the message of "Command not found"
  + PATH: colon-separated list of directories in which the shell looks
  + Common hierarchy of searching inside the ${PATH}: 'usr/gnu/bin:usr/local/bin:usr/ucb:/bin:usr/bin'
  + PATH can be changed and appended with new values.
  + "which" - not a shell built-in and shows the full path of shell commands.
    - Eg: typing 'which head' displays /usr/bin/head
    - Sudo vim /usr/local/bin/head
    - Sudo chmod /usr/local/bin/head
    - Which -a head: prints all 'head'matching executables in our path
    - Hash -r: will let shell refresh all its hash table values and locate head to its original path i.e. /usr/bin/head
  + 'basename': strips directory and suffix from filenames
    - Eg: basename /etc/passwd will give passwd
  + 'dirname': give the directory from a given file path
    - Eg: dirname /vagrant/luser-demo06.sh will give vagrant
  + Basename and dirname don’t do smart checking in the sense that, even if the file or directory doesn’t exist, it will still return the value i.e. filename or directory name
  + Command substitution: echo "you filename is $(basename ${0} and directory name is $(dirname ${0}))"
    - In the above command, we are not taking any extra variable to store the output of the basename od dirname command.
* Special Parameter: # pound sign. Expands to the number of parameters in decimal. Calculates how many arguments are passed to the command line
  + Eg: echo "The number of command line arguments are: ${#}". Sometime, people also use $# to depict the number of command line arguments
* For loop: for is a built-in shell command. It executes a sequence of commands for each member in a list of items
  + Eg:
  + for element in shell scripting class

Do

Echo "Hello ${element}"

Done

* The above command prints:
  + Hello shell
  + Hello scripting
  + Hello class
* ${@} - expands all the arguments starting from index 1. For eg: if we type ./luserdemo06.sh first second third
* It will display arguments:
  + 1: first
  + 2: second
  + 3: third

and not the 0th indexed argument i.e., luser-demo06.sh

* ${\*} - expands all the arguments starting from index 1 as one single argument. For eg: if we type ./luserdemo06.sh first second third
* It will display arguments:
  + 1: first second third
* ${\*} and ${@} are different and can be confused with. Select the appropriate one on use case basis.