***Shell Scripting***

Server IP Address –

Shell is a program which get automatically started when user logged in. There are many shell program – sh, ksh, zsh, csh, bash.

You can change login shell by command –

---------------------------------------------------------------------------------

# chsh

Password: (enter password)

$Please enter new shell (/bin/bash) : /bin/ksh

Log out and log in again.

---------------------------------------------------------------------------------

Every user in his home account or folder has ‘.bash\_profile’ file, the moment you login will automatically get executed. Whatever unix command you write in this file will be executed.

\* Files that are executed in a sequential fashion when ever user logged in –

1) /etc/profile

2) /etc/bashrc (usually alias commands, Global variables defined here)

3) .bash\_profile (From Home directory. This file is executed only your login shell is ‘bash’)

4) .bashrc (From Home directory, functions and aliases can be written here.)

* Skel directory - Super user can create profile in this “/etc/skel/” directory, and if superuser create user with ‘useradd’ command, profile will get assigned to newly created user.

Variables -

* Variable will be assigned a value with syntax like ‘a=100’.
* There should be no space before and after
* Ex. a=’100’, b=’Shekhar’
* **Use single quote (‘ ’) while assigning value of a variable (it may contains a space)**
* The value assigned will remain as it is till your session is closed.
* # set – command shows you all variables defined by system and users
* #env – Environmental variables (system defined) only, will not show user defined variables
* For print the value of variable use ‘echo’ command (echo $a)
* **While accessing a value of a variable use ‘$’ sign**
* **While assigning a value to a variable do not use ‘$’ sign**
* **The correct syntax is - echo ${a}, whenever you declare variable**
* z=$x+$y = will print value of variable x + value of variable y (ex. z=100+200)

As in unix shell scripting there is no any concept of ‘Data Type’ (integer, text, etc) and ‘Data Length’

Only one Data Type i.e. String Concatenation

Linux does not know anything like numbers, it only deals with characters.

No data types and no length restriction (The limitation is only about RAM of system)

* **Character underscore (\_) is only allowed character in the name of variable.**
* When you assigned a value to a variable no quotes are required as long as no space is there in the value
* Any written in single quote is literally printed as it is. (without having things inside it with meaning)
* Every unix command can contain another unix command in it
* ‘read' command is exactly opposite of ‘echo’ command. ‘echo’ command print the value where ‘read’ command accept the value.
* The command “ **# read –p ‘What is your name?’ nm** ” will accept the value from user and assign to variable ‘nm’ (‘-p’ option is to print that on screen )
* `echo –en \n\n ` in this command options “-en”

-e = enables interpretation of back slash escapes

-n = do not output trailing new line (by default output of echo command will go to next line)

\n = next line

\t = new tab

The above command says please do not go to next line I will tell you the new line.

* # echo -en "RAM \n\n\nSmaran\n" ---- this command will print output like this :

RAM

Smaran

* If you remove “-en” options then whatever in double quotes will print as it is.
* Just like **echo** command **printf** command is also.
* You can put **single quote** in **double quote, double quote in single quote. But you cannot use single quote in a single quote same for double quotes also.**
* If you want to copy entire content of a file into current file then you can do by typing –

:read filepath

Ex: vim ram.sh

:read sita.sh (if it is in same directory)

:read /tmp/sita.sh (if file is in /tmp directory)

* tar -cvf "${dir\_name1}".tar.gz ${dir\_name1} &>/dev/null

tar = for zipping folder (‘-cvf’ are options )

“${dir\_name1}”.tar.gz = As there is variable called, hence double quote is used here.

&> /dev/null = ‘&’ is for redirection of STDOUT and STDERR to /dev/null

‘>’ is for redirection

“/dev/null” = It acts like garbage collector (output will immediately wiped up)

1 = STDOUT

2 = STDERR

& = Both STDOUT and STDERR

Note : Do not use space in between ‘&’ and ‘>’

* Backup tack “` `” is used to write command insid command

Ex.

* Command ‘expr’ can be use for basic mathematical calculations.

Control Statements – which will control the flow of program (for loop, while loop, if statements)