**For shell script and some commands refer in : geek stuff(( very nice refer ones any linux commands are shell script loop u can refere very very nice……))**

**Programming Knowledge tutorials :**

<https://www.youtube.com/watch?v=AcSkkNAsGCY>

**Naresh technologies :**

[**https://www.youtube.com/watch?v=k4Ww6gFeF0M&t=34s**](https://www.youtube.com/watch?v=k4Ww6gFeF0M&t=34s)

**shell Tutorial Book :**

[**https://www.shellscript.sh/philosophy.html**](https://www.shellscript.sh/philosophy.html)

**##/bin/sh : (/bin/sh) tells the script to run every time with this script**

The first line tells Unix that the file is to be executed by /bin/sh. This is the standard location of the Bourne shell on just about every Unix system. If you're using GNU/Linux, /bin/sh is normally a symbolic link to bash (or, more recently, dash).

##**parameters/arguments** are nothing but **strings/words**…..parameters/arguments are passed from the **command line**….or we can write in a file and we can run those file that is called script file….

**##**echo can take any number of parameters (strings/words ,in simple manner) so we can provide or not provide double codes it s not a problem/issue.

EX : **1st case :**

Echo hello world

**2nd case :**

Echo “hello world”

## But in Variable case we must and should provide double codes..

Ex : MY\_name=”Hello World”

Echo $MY\_name

\*MY\_name is a variable and hello world is a parameters/strings

\*\*Because variable takes/holds one value.

**Shell script :**

**Shell script** is nothing but collection of Linux commands.

In every operating system shell is common or default.

At a time we cannot execute number of commands.

So we add number of commands in a file .that file is called shell.

Every script file ends with .sh extension and .sh is not mandatory..

Sh is nothing but standard shell .this is popularly known as boune shell.

If we want know what shell is currently running in our system we hit echo $0 in command prompt.

If want to convert to the another shell suppose ksh …we want to hit ksh in command prompt..and if we want to know which shell is running we give again echo $0

Exit is the command to exit from current shell to another…

Commonly we are using bash and bash prompt is $ and bash prompt syntax is sh [standard shell]

#uses of shells :

1.Customizing our work environments

2.automating our daily tasks

3.Automating repetitive tasks

4.Exectuing important system procedure like shutting down the systems.

5.performing same operating on many files.

#types of shells :

There are diiifferent types of shells :

1. The Bourne Shell
2. The korn shell
3. The C shell

#Bourne shell :

The original linux shell is known as sh, short for shell or the Bourne Shell, the shell prompt is $ Executing on command sh.

The C shell :

C shell was so named because much of its syntax parallels that of the c programming language the shell prompt is % and exectution command is Csh

#Korn shell :

Shell prompt is $ and executable on command is ksh

@Responsibilities of the SHELL :

The shell is ultimately responsible for making sure that any commands typed at the prompt get properly executed. The following are the responsibilities of a shell

1.Program Execution

2.variable and the file Substitution

3.I/O redirection

4.Pipeline Hookup

5.Environment CONTROL

6.interpreted Prrogramming languages

**#Shell variables :**

Variable is data name and it is used to store value. Variable value can change during execution of the program.. because echo will take any number of parameters, a variable can only hold one value,  Variables are two types :

1.System defined variables

2.User defined Variables

Perl means practically extraction report language.

#1.System defined variables :

Environmental variables are used to provide information to the programs you use . you can have both global environment and local shell variables. Global environment variables are set by our login shell and new programs and shells inherit the environment of their parent shell

@some common ones are :

1.DISPLAY : The graphical display to use

2.EDITOR : the path to your default editor

3.GROUP : our login group

4.HOME : path to our home directory

5.HOST : the hostname of our system

6.IFS : internal field separator [tab/space/enter etc..]

7.LOGNAME : the name u login with

8.PATH : paths to be searched for commands

9.PS1 : the primary prompt string [$]

10.PS2 : secondary prompt string [default to > ]

Export is the command to export globally…suppose

For example :

If we declare x=33

Echo $x then it displays 33in those shell only

If we exit from those shell it will not display because it is not set as globally.so to make globall we use export command.

@userdefined variables :

These are defined by user and are used most extensively in shell programming.

Creating user defined variables

1.The first character of a variable name should be alphabet or underscore.

2.No commas or blanks are allowed within a variable name

3.variables names should be of any reasonable length

4.variable names are case case sensitive

5.It shouldn’t be reserved word.

#shell keywords :

1.echo

2.if read else set if unset until trap case wait done esac eval break exec continue read-only while do ulimit shift exit umask

##Mainly shell script is classified into two types :

1. Interactive shell script an
2. Non interactive shell script

1.Interactive shell script :

Means while script is running it takes any input from user that is called interactive

2.Non interactive :

Means if a script is executing it doesn’t take any value from the user that is called non interactive.

Users : print the user names of users currently logged in to the current‐host

Who : show who is logged on

#Display the domain name

Hostname –d

#display ip address

Hostname –i

#dispaly current week number

Date +%u

#Display linux flavur

Uname –s

#dispaly name of the processor

Uname –p

#display date in yyyy/mm/dd

Date +%F

#dispaly day number in a year

Cal –j

##.sh is not compulsory it is optional….

**Read** : by using read command we can take input from the user…input statement/input command

Ex : read a

Raju

Echo $a

Raju

#Integer comparison :

-eq : is equal to if [ $a –eq $b ]

-ne

-gt

-ge

-lt

-le

< : is lessthan if (( $a <= $b ))

<=

>

>=

##String comparison :

= : is equal to if [ $a = $b ]

== : is equal to if [ $a == $b ]

!= : not equal to if [ $a != $b ]

< : is less than

> : is greater than

-z : string is null , that is , has zero length

##if u want to find if a file exist or not then

-e : exist

For example :

Read $file

If [ -e $file ]

-f : if it is a regular file

-d : if it is a directory

-b : if it is a block special file is a binary file that means if it is a picture file or video file .

-c : is a character file which contains a text.

-s : it checks if a file is empty.

&& : if we want to write more than one condition we use it..

For Ex :

If [ 25 –gt 24 ] && [ 25 –lt 30 ]

Or we can use

If [ 25 –gt 24 -a 25 –lt 30 ]

-a is also called and..

And also we can use :

If [[ 25 –gt 24 && 25 –lt 30 ]]

##logical “or/-o”

Is used ,if we are writing more then two conditions..

For Ex :

If [[ 25 –gt 24 -o 25 –lt 30 ]]

Or

If [ 25 –gt 24 ] | | [ 25 –lt 30 ]

Or

If [[ 25 –gt 24 | | 25 –lt 30 ]]

##Types of brackets :

parentheses or "round brackets" ( )

"square brackets" or "box brackets" [ ]

braces or "curly brackets" { }

"angle brackets" < >

##Arthimetic operations :

For ex :

Num1=20

Num2=25

Echo $( ( $num1 + num2 )) = addition

Echo $( ( $num1 - num2 )) =substraction

Echo $( ( $num1 /\* num2 )) =multiplication

Echo $( ( $num1 % num2 )) =modulus

Echo$( ( $num1 / num2 )) =division

Or

Echo $( exp $num1 + num2 ) = addition

Echo $( exp $num1 - num2 )=substraction

Echo $( exp $num1 \\* num2 )=multiplication

Echo $( exp $num1 % num2 )=modulus

Echo $( exp $num1 / num2 )=division

##basic calculator :

We use this command because 0.5 [after . the remaining numbers are called decimal numbers]that means decimal numbers will not support for

$( ( $num1 + num2 )) this method so we have to

Use basic calculator[bc]….

For Ex :

Echo $num1 + $num2 | bc

Echo $num1 - $num2 | bc

Echo $num1 \* $num2 | bc

Echo $num1 / $num2 | bc

Echo $num1 % $num2 | bc

And also we can use scale command if we want we can refer in **man bc**

For ex :

**echo scale=2;$num1+$num2 | bc**

above one gives us after .[point] two decimal number

Ex : suppose 20.25 if we give scale=5 we get 20.25436

Please check manual page for bc for options ……

##case statement :

**Regular Expressions :**

. [ ] [^] ^ $ ( ) \n \* {m,n} \* (star)is called asterisk

For complete reference

<https://en.wikipedia.org/wiki/Regular_expression>

##Array variables :

Suppose if we assign a value to a variable like

String=asphalt

Echo ${string[0]}

\*It prints asphalt because it takes first index/element as o and if we give

String=(asphalt mass down)

Echo ${string[1]}

\*it prints mass because it takes second element as 1 and third element as second

String=(“asphalt” “ mass” “down”)

Echo ${string[@]}

\*it prints asphalt mass down because double couted elements are equal to $@..

String=(asphalt mass down)

Echo ${string[\*]}

\*it prints asphalt mass down because all the elements are equal to $\*

A **shell** variable can store only a single value. These variables are called **scalar** **variables.** **Shell** also supports a different type of variable called an **array** **variable**. This can hold multiple values at the same time. .

Ex :

Scalar=seena

Defining=$scalar

Ex :

Array[0]=seena

Array[1]=Chandra

Defining = ${array[@]}

**Egrep :**

**egrep "^ab|^xy" login.java**

searching with first two charancters

**egrep "ab$|xy$" login.java**

searching with last two characters