Shell

-----

What is shell?

Shell is a small program, which will help a user to interact with operating system (OS).

Steps of processing in execution at OS?

Terminal (or) application

|

Shell

|

OS

|

Kernel

|

Hardware

What are different types of shells?

/bin/sh ---------> bourne shell

/bin/bash -------> advance bourne shell

/bin/csh --------> C shell

/bin/tsh --------> advance C shell

/bin/ksh --------> korn shell

/bin/fish -------> fish shell

Every shell is same but the difference is only syntax at shell script.

Ex: #bash script

If [ condition ];then

# Block

fi

#C script

if(condition)

{

#block

}

What is shell script?

It is a collection of commands executed in a given order with some logic.

Can I execute bash script in c shell?

No

How can I write my shell script?

The first line of script is always represents the shell.

Ex:

#!<path of shell>

#!/bin/bash ----> it represents the bash shell

#! ---->shebang operator

It tells to the os that we need to execute this script by this shell.

What is comment in shell script?

Shell does not execute the comment line. Comment line starts with '#' symbol.

NOTE: Except the first line.

What is my extension of shell script?

<scriptname>.sh ------> recommended for bash script

<scriptname>.csh -----> c shell

<scriptname>.ksh -----> korn shell

NOTE: without extension of also the script is valid.

<scriptname>

Because your first line tells to the os which shell is execute this script.

Extensions for human understanding purpose only.

What is my script structure?

#!/bin/bash

#commands

#commands

---------

---------

How to execute shell script?

There are 3 methods to execute the script.

Method 1

--------

bash <scriptname>.sh

(Or)

bash <scriptname>

Method 2

--------

sh <scriptname>.sh

(Or)

sh <scriptname>

Method 3

--------

Go to the script directory and

./<scriptname>.sh

(Or)

./<scriptname>

To execute in this way you need to give the executable permissions to that script.

chmod +x <scriptname> ----> give all permissions to that script to all

chmod 755 <scriptname> ---> give execute permissions to that script to all.

Shell variables

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

What is variable?

It is a label (or) name given to a memory location. Which may hold some value.

x=9

X--->it is a variable name

Why we need a variable?

To store a value temporally or permanently

What are different types of variables?

There are 3 different types of variables.

1) System (or) environment variables --> $HOME,$PATH,$EDITOR,$HOST,etc....

2) User defined variables -----> x, y,age,name,etc...

3) Shell special variables ---> $0,$1,$@,$#,$$,$\*

How to see environment variables?

cat /env | less

How to print the variable data on to screen?

Using "echo" command we can print the data on to the screen.

How to see individual environment variables?

echo $HOME

How to define a variable name?

<varname>=<value>

age=21

echo $age -->to print the data of that variable

What name is use to my variable?

For using a variable name use the valid name only

Valid names

Ex: x="technicaladda"

x\_name="technical adda"

\_x="technical adda"

x1="technical adda"

NOTE: special characters are not allowed in variable name. And variable name does not starts with numbers.

Ex: 1X="hello"

x@5="world"

(Wrong variable format)

How to assign an output of command to the variable?

<Varname>=`command`

We place the command in `` (back tick)

name='hostname' ----> hostname is assign to the name variable.

How to read the data from user?

Using "read" command we can read the data from user.

Ex:

read x

echo $x

read "what is your name?" name

echo "my name is $name"

read "enter your age?" age

echo "my age is $age"

Operators:

---------

= -->assign operator

Athematic operators

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

+ -->addition

- -->subtraction

\* -->multiplication

/ -->division

% -->reminder

NOTE: in shell we are not directly perform the athematic operations use the "expr" command to perform this

Ex: c=`expr $a+$b`

(Or)

c=$((a+b))

Comparison operators

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

-eq (==) -->equal operator

-ne (!=) -->not equal

-gt (>) -->greater than

-ge (>=) -->greater than or equal

-lt (<) -->less than

-le (<=) -->less than or equal

Logical operators

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

-a (and) -->logical and

-o (or) -->logical or

! (not) -->not operator

Escape sequences

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

\n ->to go the next line

echo "my name "ramu"" ---------> syntax error

echo "my name is \"ramu\"" ------>o/p my name is "ramu"

Conditional statements:

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

Conditional statements are control the flow of program.

There are different conditional statements.

1) if..fi -->if block

2)if..else..fi -->if else block

3)if..elif..elif..else..fi -->else if ladder

What is condition?

Condition is an expression which will either true or false.

If:

---

If statements are used to make a decision. Decision means condition, if the condition is valid then execute this block otherwise not allowed to come in block.

if syntax

---------

if [ condition ];then --->if statement start

#true block

#commands

fi ---------------> if statement end

if..else

--------

if..else will give if condition is true if block is executed,otherwise else block is executed.

if..else syntax

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

if [ condition ];then --->if statement start

#true block

#commmand

else

#false block

#commads

fi ---------------> if statement end

if..elif..elif..else

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

this statement will give more choices to the user.

syntax

------

if [ condition ];then --->if statement start

#true block

---------

elif [ codition ];then

#true block

-----------

elif [ condition ];then

#true block

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

else

#false block

-----------

fi ---------------> if statement end

here we can see the different types of tests using if statements.8

1)file test

2)number test

3)string test

4)logical test

1)file test

-----------

using this test we can perform operations on file

some options are their to perform operations on file

-f -->check if the file exists or not

-d -->check if the dir exists or not

-r -->check if the file/dir has read permissions or not

-w -->check if the file/dir has write permissions or not

-x -->check if the file/dir has executable permissions or not

-s -->check if the file/dir size is greater than zero or not

if [ -f "/etc/passwd" ];then

echo "passwd file is exists"

fi

if [ -r "/etc/shadow" ];then

echo "shadow file has read permissions"

else

echo "shadow file does not have read permissions"

fi

2)string test

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

we can check the strings equal or not

str1==str2

str1!=str2

3)number test

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

x==y

x!=y

x>y

x>=y

x<y

x<=y

marks=60

if [ $marks -gt 70 ];then

echo "grade A"

elif [ $marks -gt 60 ];then

echo "grade B"

elif [ $marks -gt 40 ];then

echo "grade C"

elif [ $marks -le 40 ];then

echo "fail"

else

echo "invalid input"

fi

what is passing value by argument?

it is one way to send the arguments as input to the script.so,here while script is going to be run we pass the arguments as input from the command line.

from the command line we can pass max 9 arguments.

./<scriptname>.sh <arg1> <arg2> ..............<arg9>

the argument names are $0,$1...............$9

$0 represents script name.

4)logical test

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

-a (and) | -o (or) | ! (not)

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

A B o/p | A B o/p | A B

----------------------------------------------------------------------------- | |

T T T | T T T | T F

T F F | T F T | F T

F T F | F T T |

F F F | F F F |

ex: if [ ! $1 ];then

echo "argument required"

if [ $marks -gt 70 -a $marks -le 60 ];then

echo "grade A"

elif [ $marks -lt 60 -a $marks -ge 50 ];then

echo "grade B"

elif [ $marks -le 50 -a $marks -gt 40 ];then

echo "grade C"

if [ $marks -le 40 -a $marks -gt 0 ];then

echo "fail"

else

echo "invalid input"

fi

home work 1

-----------

write a script to create a user?

1)take the username as command line argument

2)set the default password

3)check you have privilages to create user

4)check the user is already present or not

5)create a user

6)assign a password

7)make a user as admin user

case statement

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

case statement is give multiple choices to the user.

syntax

-----

case <varname> in --->case statement start

choice1)#1st action ;;

choice2)#2nd action;;

choice3)#3nd action;;

\*)#if all matches not found

esac --->case statement end

ex: case $osname in

centos)echo "you choose centos"

echo "use yum to install the packages"

ubuntu)echo "you choose ubumtu"

echo "use apt-get to install the packages"

solaris)echo "you choose solaris"

echo "use package to install the packages"

\*)echo "you choose wrong osname"

home work 2

-----------

convert case statement to if..elif..elif.else statements.

loops

-----

loops are executed a block of the code repeatedly in a specific number of times.

there are different types of loops are their.

1)for loop

2)while loop

3)until loop

1)for loop

-----------

to execute the block of the code repeatedly in spectific tiemes.

syntax

------

for <varname> in <data (or) list>

do -->block start here

#block of code

#action here

done -->block end here

create the 100 numbers in a single step

ex:for i in `seq -w 1 1 100`

do

echo $i

done

ex:for i in windows linux solaris mac

do

echo "the os name is $i"

done

home work 3

-----------

create multiple users using for loop.

while loop

----------

a block of code is executed repeatedly until condition fails.

syntax

------

initilazer

while [ condition ]

do -->block start here

#block of code if the condition satisfies then enter to this block

#modifier

done -->block end here

ex:i=1

while [ $i -le 10 ]

do

echo $i

i=`expr $i+1`

(or)

i=$((i+1))

homework 4

----------

take a backup of etc dir daily.

etc-04-oct-2017.tar.gz

etc-05-oct-2017.tar.gz

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

until loop

----------

a block of code is executed repeatedly until condition true.

syntax

------

initilizer

until [ condition ]

do ---->block start here

#action here if condition fails then execute this block

done ---->block end here

ex:

i=1

until [ $i -gt 10 ]

do

echo $i

i=`expr $i+1`

done

functions:

----------

what is a function?

function is a peace of code to do a particular task.

why we need function?

1)to avoid repeat of code

2)make it simple and efficient

syntax

------

function() -->function defination

{

#action

}

function -->function call

what is function defination?

function defination describes the what the function can do.

ex:add()

{

count=1

sum=0

while [ $count -le 10 ]

do

sum=`expr $count+$sum`

done

echo "the sum of numbers is $sum"

}

what is function call?

just we call the function with function name,after calling happen only the function is executed.

add ---->just specify the name to call the function

can we pass the arguments to the function?

yes

how can we pass the arguments to the function?

add()

{

echo `expr $0+$1`

}

add 4 3 -->passing 4 3 to the add function

can function return anything?

yes,using "return" key word it can return

add()

{

return `expr $0+$1` -->return the addition value

}

sum=add 4 3 -->return value store in sum variable

echo $sum

fumction file

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

what is the speciality of function file?

1)it doesnot start with #! (shebang operator)

2)it will have execute permission

3)it has to load in memory.

in the function file just we define the functions task only.

just load this file in to the memory.

ex:showdir()

{

ls -l | grep "^d"

}

we make shortcuts using functions.

how to load functions in memory?

source <function file name>

how to unload function from memory?

unset <function name>

we can not unload all functions at a time. to unload all the functions from memory just logout and login.