Exp No.3 Date:

Shell script programming

Aim:

Introduction to shell script programming

Introduction

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.

ECHO COMMAND:

When we use '**echo**' command without any option then a newline is added by default. **'-n'** option is used to print any text without new line and **'-e'** option is used to remove backslash characters from the output.

```
echo "Printing text with newline"
echo -n "Printing text without newline"
echo-e "\nRemoving \t backslash \t characters\n"
```

Use of comment:

‘#’ symbol is used to add single line comment in bash script.
‘:’ and **“ ’ ”** symbols are used to add multiline comment in bash script.

Get User Input:

‘**read**’ command is used to take input from user in bash. Here, one string value will be taken from the user and display the value by combining other string value.

```
#!/bin/bash
echo "Enter Your Name"
read name
echo "Welcome $name to LinuxHint"
```

Using if statement:

You can use if condition with single or multiple conditions. Starting and ending block of this statement is define by **‘if’** and **‘fi’**.

```
#!/bin/bash
n=10
if [ $n -lt 10 ];
then
echo "It is a one digit number"
else
echo "It is a two digit number"
fi
```

Here, **10** is assigned to the variable, **n**. if the value of **$n** is less than 10 then the output will be "**It is a one digit number**", otherwise the output will be "**It is a two digit number**". For comparison, **'-lt'** is used here. For comparison, you can also use **'-eq'** for **equality**, **'-ne'** for **not equality** and **'-gt'** for **greater than** in bash script.

Using if statement with AND logic:

Different types of logical conditions can be used in if statement with two or more conditions. **'&&'** is used to apply **AND** logic of **if** statement.

```
!/bin/bash

echo "Enter username"
read username
echo "Enter password"
read password

if [[ ( $username == "admin" && $password == "secret" ) ]]; the
n
echo "valid user"
else
echo "invalid user"
fi
```

Here, the value of **username** and **password** variables will be taken from the user and compared with '**admin**' and '**secret**'. If both values match then the output will be “**valid user**”, otherwise the output will be “**invalid user**”.

Using if statement with OR logic:

‘||’ is used to define **OR** logic in **if** condition.

```
#!/bin/bash

echo "Enter any number"
read n

if [[ ( $n -eq 15 || $n  -eq 45 ) ]]
then
echo "You won the game"
else
echo "You lost the game"
fi
```

Here, the value of **n** will be taken from the user. If the value is equal to **15** or **45** then the output will be “**You won the game**”, otherwise the output will be “**You lost the game**”.

Using else if statement:

‘**elif**’ is used to define **else if** condition in bash.

```
#!/bin/bash

echo "Enter your lucky number"
read n

```

```
if [ $n -eq 101 ];
then
echo "You got 1st prize"
elif [ $n -eq 510 ];
then
echo "You got 2nd prize"
elif [ $n -eq 999 ];
then
echo "You got 3rd prize"

else
echo "Sorry, try for the next time"
fi
```

Using While Loop:

```
#!/bin/bash
valid=true
count=1
while [ $valid ]
do
echo $count
if [ $count -eq 5 ];
then
break
fi
```

```
((count++))
done
```

In the example, **while** loop will iterate for **5** times. The value of **count** variable will increment by **1** in each step. When the value of **count** variable will 5 then the **while** loop will terminate.

Using For Loop:

for loop will iterate for **10** times and print all values of the variable, **counter** in single line.

```
#!/bin/bash
for (( counter=10; counter>0; counter-- ))
do
echo -n "$counter "
done
printf "\n"
```