OSL Assignment 1A

Aditya Kangune

K 11

33323

#Study of basic Linux commands

1. echo:

The *echo* command is used to display a line of text.

Syntax:

echo [SHORT-OPTION] ... [STRING]...

Description(option):

-n do not output the trailing newline

-e enable interpretation of backslash escapes

-E disable interpretation of backslash escapes (default)

If -e is in effect, the following sequences are recognized:

\\ backslash

\a alert (BEL)

\b backspace

\c produce no further output

\e escape

\f form feed

\n new line

\r carriage return

\t horizontal tab

\v vertical tab

Example:

adi@adi-VirtualBox: $ echo "Hello World, welcome!"

Hello World, welcome!

adi@adi-VirtualBox:$ echo -e "Hello\t\tWorld"

Hello World

2. ls

The *ls* command is used to list directory contents.

Syntax:
ls [OPTION]... [FILE]...

Description(option):

-i print index number of each file

-l use long listing format

-t sort by modification time, newest first

-1 list one file per line.

Example:

adi@adi-VirtualBox:~$ ls -1

Desktop

Documents

Downloads

Music

OpenGLExample

Pictures

Public

Sample-OpenGL-Programs

Templates

Videos

3. read

The *read* command is used to read the contents of a line into a variable. It is primarily used for catching user input.

Syntax:
read [variable]

Example:

adi@adi-VirtualBox:~$ read fname sname
Aditya Kangune
adi@adi-VirtualBox:~$ echo "Name: $fname $sname"
Name: Aditya Kangune

4. cat

The *cat* command is used to Concatenate files and print on the standard output.

Syntax:
cat [OPTION] [FILE]

Description(option):
-E: display $ at end of each line

-n: number all output lines

-b: number nonempty output lines, overrides -n

Example:

adi@adi-VirtualBox:~ $ cat -n demo.sh

```
1	#!/bin/bash
2	echo "Enter name: "
3	read fname sname
4	echo "Name: $fname $sname"
5
6	read -p 'username:' user_val
7	echo "username: $user_val"
8
9	echo "Enter name:"
10	read -a names
11	#print names
12	echo "Name array has: ${names[0]}, ${names[1]}, ${names[2]}"
13
14	echo "Enter password:"
15	read -s password
16	echo "Password: $password"
17
18	#testing variables
19	days=7
20	guest="Aditya"
21	echo "$guest checked in $days ago"
```

adi@adi-VirtualBox:~$ bash demo.sh

Enter name:

Aditya Kangune

Name: Aditya Kangune

username:adityaubuntu

username: adityaubuntu

Enter name:

Ruturaj Yash Kshitij Aditya Yash

Name array has: Ruturaj, Yash, Kshitij, Aditya

Enter password:

Password: ubuntu

Aditya checked in 7 ago

5. touch

The *touch* command is used to create, change, and modify timestamps of a file

Syntax:

touch [OPTION] [file name]

Description(option):

-c: used to check whether a file is created or not.

-m: used to change the modification time.

Example:

adi@adi-VirtualBox:~$ ls
Documents Music Pictures Sample-OpenGL-Programs Templates
Desktop Downloads Public snap Videos

adi@adi-VirtualBox:~$ touch newFile.txt

adi@adi-VirtualBox:~$ ls

Documents Music Public snap Videos Desktop Downloads newFile.txt Pictures Sample-OpenGL-Programs Templates

6. test

The *test* command is used to check file types and compare values. Test is used in conditional execution.

Syntax:

Test EXPRESSION

Description:

Test exits with the status determined by EXPRESSION. To see the exit status at the command prompt, echo the value "$?" A value of 0 means the expression evaluated as true, and a value of 1 means the expression evaluated as false.

Example:

adi@adi-VirtualBox:~ $ test 100 -gt 99; echo $?

0

adi@adi-VirtualBox:~$ test "a" = "b"; echo $?

1

7. grep

The *grep* command is used to search text and strings in a given file.

Syntax:
grep [-options] pattern filename

Description(option):

-c: print count of the lines that match the pattern.
-h: display the matched lines, but do not display the filenames.

-i: Ignores, case for matching
-l: Displays list of a filenames only.
-n: Display the matched lines and their line numbers.
-v: This prints out all the lines that do not matches the pattern

Example:

adi@adi-VirtualBox:~$ cat newFile.txt
Hi.
How are you?
I'm ok.

adi@adi-VirtualBox:~$ grep -i HoW newFile.txt
How are you?

8. Arithmetic Expressions(expr):

The *expr* command is used to evaluate arithmetic expressions.

Syntax:
expr EXPRESSION

Description:
Expression can be:
ARG1 < ARG2: ARG1 is less than ARG2.
ARG1 <= ARG2: ARG1 is less than or equal to ARG2.
ARG1 = ARG2: ARG1 is equal to ARG2.
ARG1 != ARG2: ARG1 is unequal to ARG2.
ARG1 >= ARG2: ARG1 is greater than or equal to ARG2.

Example:

adi@adi-VirtualBox:~$ cat -n demo2.sh

```
1	#!/bin/bash
2	x=3;y=5
3	expr 3 + 5
4	expr 3 / 5
```

```
5       expr $x - $y
6       expr $y % $x
7
8       z=`expr $x + $y`
9       echo "z= $z"
10      a=`expr $x + 1`
11      echo "x on incrementing = $a"
```

adi@adi-VirtualBox:~$ bash demo2.sh

8

0

-2

2

z= 8

x on incrementing = 4

9. Loops

a. For loop:

The *for* loop executes a sequence of commands for each member in a list of items.

Example:

adi@adi-VirtualBox:~$ cat for_loop.sh

```
#!/bin/bash
#for i in 1 2 3 4 5
#do
#echo "Printing looping....number $i"
#done

#for i in {1..10}
#do
```

```
#echo "i= $i"
#done

#for i in {1..10..2}
#do
#echo "i= $i"
#done

for (( j=0; j<5; j++))
do
echo "j= $j"
done
```

adi@adi-VirtualBox:~$ bash for_loop.sh

```
j= 0
j= 1
j= 2
j= 3
j= 4
```

b. While loop:

The *while* loop is used to execute commands as long as a test succeeds.

Example:

adi@adi-VirtualBox:~$ cat while_loop.sh

```
#!/bin/bash

n=1
while [ $n -le 10 ]
do
echo "n= $n"
n=$(( n+1 ))
done
```

adi@adi-VirtualBox:~$ bash while_loop.sh

```
n= 1
n= 2
n= 3
n= 4
```

```
n= 5
n= 6
n= 7
n= 8
n= 9
n= 10
```

adi@adi-VirtualBox:~$ cat while_1.sh

```
#!/bin/bash

a=0
while [ "$a" -lt 10 ]
do
    b="$a"
    while [ "$b" -ge 0 ]
    do
            echo -n "$b" b=`expr $b`
            b=`expr $b - 1`
    done
    echo
    a=`expr $a + 1`
done
```

adi@adi-VirtualBox:~$ bash while_1.sh

```
0 b=0
1 b=10 b=0
2 b=21 b=10 b=0
3 b=32 b=21 b=10 b=0
4 b=43 b=32 b=21 b=10 b=0
5 b=54 b=43 b=32 b=21 b=10 b=0
6 b=65 b=54 b=43 b=32 b=21 b=10 b=0
7 b=76 b=65 b=54 b=43 b=32 b=21 b=10 b=0
8 b=87 b=76 b=65 b=54 b=43 b=32 b=21 b=10 b=0
9 b=98 b=87 b=76 b=65 b=54 b=43 b=32 b=21 b=10 b=0
```

adi@adi-VirtualBox:~$ cat while_2.sh

```
#!/bin/bash

a=0
while [ $a -lt 10 ]
do
```

```
    b=$a
    while [ "$b" -ge 0 ]
    do
            echo -n "$b"
            b=`expr $b - 1`
    done
    echo
    a=`expr $a + 1`
done
```

adi@adi-VirtualBox:~$ bash while_2.sh

```
0
10
210
3210
43210
543210
6543210
76543210
876543210
9876543210
```

c. Until Loop:

The *until* loop is used to execute commands as long as a test does not succeed.

Example:

adi@adi-VirtualBox:~$ cat until.sh

```
#!/bin/bash

 n=1
 until [ $n -ge 10 ]
 do
    echo $n
    n=$(( n + 1 ))
done
```

adi@adi-VirtualBox:~$ bash until.sh

1
2
3
4
5
6
7
8
9

10. sed

The *sed* command is used to perform basic operations on an input stream such as a file. sed is a stream editor for filtering and transforming text

Syntax:

sed [option] [input file]

Description:

Option:

-i: Edit files in place.

-n: suppress automatic printing of pattern space.

Commands:

/c text: Replace the selected lines with text.

/d: Delete pattern space.

Example:

adi@adi-VirtualBox:~$ cat newfile.txt

Hi how are you?

My name is aditya

I live in pune

adi@adi-VirtualBox:~$ sed -i "/Hi/d" newfile.txt

adi@adi-VirtualBox:~$ cat newfile.txt

My name is aditya

I live in pune

11. case:

Example:

adi@adi-VirtualBox:~$ cat case.sh

```
#!/bin/bash

vehicle=$1

case $vehicle in
   "car" )
      echo "Rent of $vehicle is 100 dollars" ;;
   "van" )
      echo "Rent of $vehicle is 80 dollars" ;;
   "bicycle" )
      echo "Rent of $vehicle is 5 dollars" ;;
   "truck" )
      echo "Rent of $vehicle is 150 dollars" ;;
   * )
      echo "Unknown Vehicle" ;;

esac
```

adi@adi-VirtualBox:~$ bash case.sh car

Rent of car is 100 dollars

adi@adi-VirtualBox:~$ bash case.sh truck

Rent of truck is 150 dollars

adi@adi-VirtualBox:~$ bash case.sh

Unknown Vehicle

adi@adi-VirtualBox:~$ cat case_2.sh

```
#!/bin/bash

echo -e "Enter some character : \c"
read value

case $value in
   [a-z] )
      echo "User entered $value between a to z" ;;
   [A-Z] )
      echo "User entered $value between A to Z" ;;
   [0-9] )
      echo "User entered $value between 0 to 9" ;;
   ? )
      echo "User enter $value special character" ;;
   * )
      echo "Unknown input" ;;

esac
```

adi@adi-VirtualBox:~$ bash case_2.sh

Enter some character : 5

User entered 5 between 0 to 9

adi@adi-VirtualBox:~$ bash case_2.sh

Enter some character : p

User entered p between a to z

adi@adi-VirtualBox:~$ bash case_2.sh

Enter some character : Z

User entered Z between A to Z

adi@adi-VirtualBox:~$ bash case_2.sh

Enter some character : $

User enter $ special character