Lab No. 12
Writing Shell Scripts using Conditional-Statements, and Loops

Objective
The objective of this lab is to familiarize the students with the use of conditional and looping statements in shell scripting.

Activity Outcomes:
On completion of this lab students will be able to

- Write shell scripts that uses conditional statements
- Use looping statements available in bash shell

Instructor Notes
As pre-lab activity, read Chapter 27 &31 from the book "The Linux Command Line", William E. Shotts, Jr.

1) Useful Concepts

Conditional Statements

A conditional statement tells a program to execute an action depending on whether a condition is true or false. It is often represented as an if-then or if-then-else statement.

if Statement
if statement is provided in many variant in bash shell. The most commonly used syntax is as given below:

```
if  [ condition ]
      then
          commands
fi
```

Note that there must be a space between condition and both opening and closing brackets. The syntax of else-if command is

```
if  [ condition ]
then
      commands
elif [ condition ]
```

```
then
          commands
.
.
.
else
             commands
fi
```

The following example shows the use of if statement. In this example, we take a number as input from user and check whether it is even or odd.

```
#! /bin/bash
echo "Please Enter a Number"
read  num
if [ $((num%2)) -eq 0 ]
then
          echo "$num is an even
number"
else
           echo "$num is an odd
number"
fi
```

Exit Status

Commands issue a value to the system when they terminate, called an exit status. This value, which is an integer in the range of 0 to 255, indicates the success or failure of the command's execution. By convention, a value of zero indicates success and any other value indicates failure. The exit status of a command is saved in a system variable $?.

Other syntax for if statement

Recent versions of bash include a compound command that acts as an enhanced replacement for test. It uses the following syntax:

```
if  [[ condition ]]
then
           commands
fi
```

Similarly, (( )) syntax can also be used which is designed for arithmetic expressions.

Case statement

The bash case statement is generally used to simplify complex conditionals when you have multiple different choices. Using the case statement instead of nested if statements will help you make your bash scripts more readable and easier to maintain. The syntax of case statement is:

```
case    EXPRESSION
in
Pattern-1)

Commands
                ;;
Pattern-2)

Commands
               ;;
          .
          .
          .
Pattern-n)

Commands
              ;;
*)

Commands
              ;;
esac
```

Following are some examples of some valid patterns for the case statement.

Pattern	Description
a)	Matches if *word* equals "a".
[[:alpha:]]	Matches if *word* is a single alphabetic character.
???)	Matches if *word* is exactly three characters long
***.txt)**	Matches if *word* ends with the characters ".txt".
***)**	Matches any value of *word*. It is good practice to include this as the last pattern in a case command, to catch any values of *word* that did not match a previous pattern; that is, to catch any possible invalid values.

Looping Statements

A program loop is a series of statements that executes for a specified number of repetitions or until specified conditions are met. While, until and for are the common looping statement provided by bash shell.

For Loop

The original syntax of for loop is:

```
for     variable    in
values
do
        statements
```

```
done
```

Following example shows the working of for loop

```
#!/bin/bash
for     i   in  A B C
D E
do
        echo $i
done
```

Out-Put
A
B
C
D
E

We can give a range of values as {0..9}. bash shell also supports a C like syntax of for loop which is given below:

```
for  (( Expression;  Expression2; Expression))
do
            statements
done
```

While Loop

The syntax of while loop is as given below

```
While (( Condition ))
do
            statements
done
```

Following example shows the working of while loop

```
#!/bin/bash
count=1
while [[ $count -le 5 ]]
do
    echo $count
    count=$((count + 1))
done
```

Out-Put
1
2
3
4
5

Breaking the while loop: bash provides two built-in commands that can be used to control program flow inside loops. The break command immediately terminates a loop, and program control resumes with the next statement following the loop. The continue command causes the remainder of the loop to be skipped, and program control resumes with the next iteration of the loop.

Until Loop

The until command is much like while, except instead of exiting a loop when a nonzero exit status is encountered, it does the opposite. An until loop continues until it receives a zero exit status i.e. the condition becomes true.

```
#!/bin/bash
count=1
until [[ $count -gt 5
]]
do
  echo $count
```

Out-Put
1
2
3
4

count=$((count + 1)) done	**5**

2) Solved Lab Activities

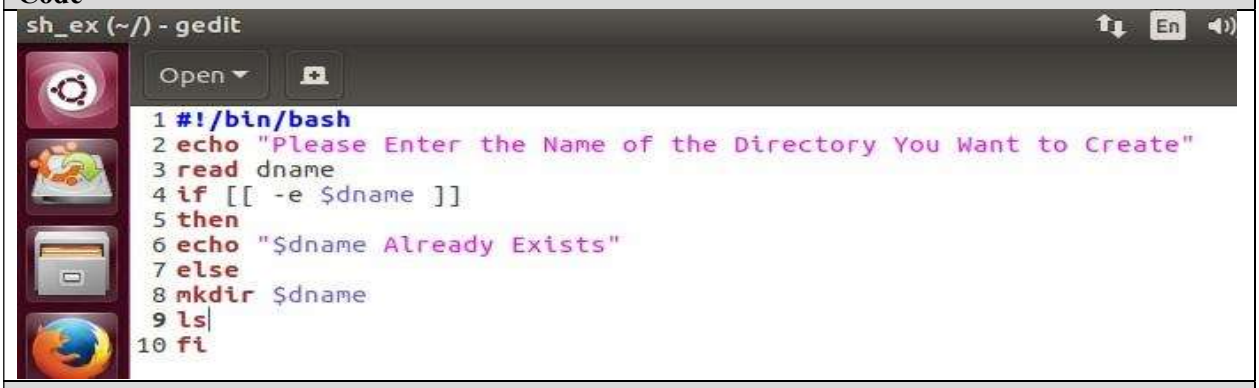
Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
1	*10*	*Low*	*CLO-6*
2	*15*	*Medium*	*CLO-6*
3	*15*	*High*	*CLO-6*

Activity 1:

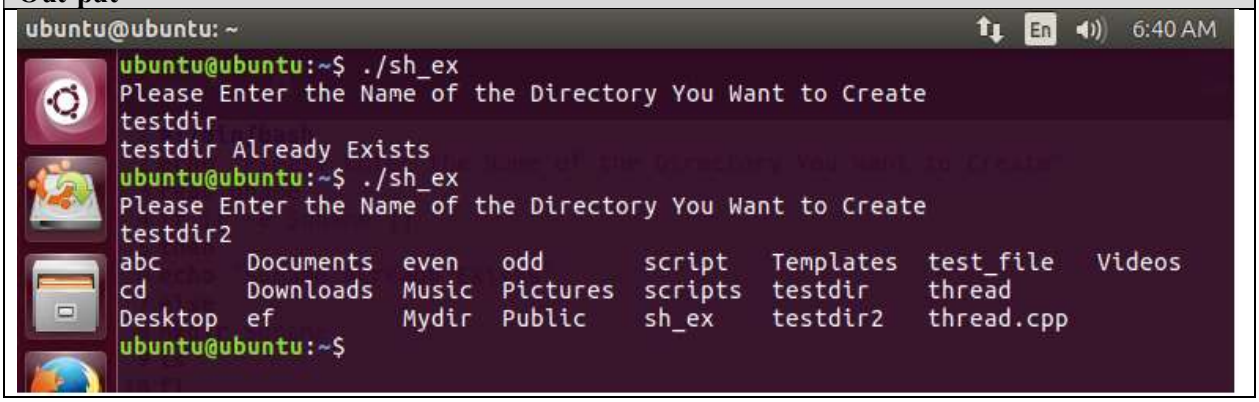
Write a shell script that modifies the mkdir command as: first it takes the directory name from the user as input and checks if a directory with same name exists then shows an error message otherwise creates the directory and show the success message.

Solution:

Code



Out-put



Activity 2:

Write a shell script that takes a word as input and finds whether it is a single alphabet, ABC followed by a digit, of length 3, ends with .txt or it is something else.

Solution:

Code	**Out-put**
#!/bin/bash read -p "enter word > " case $REPLY in [[:alpha:]]) echo "is a single alphabetic character." ;;	**Out-Put** **1** **2** **3**

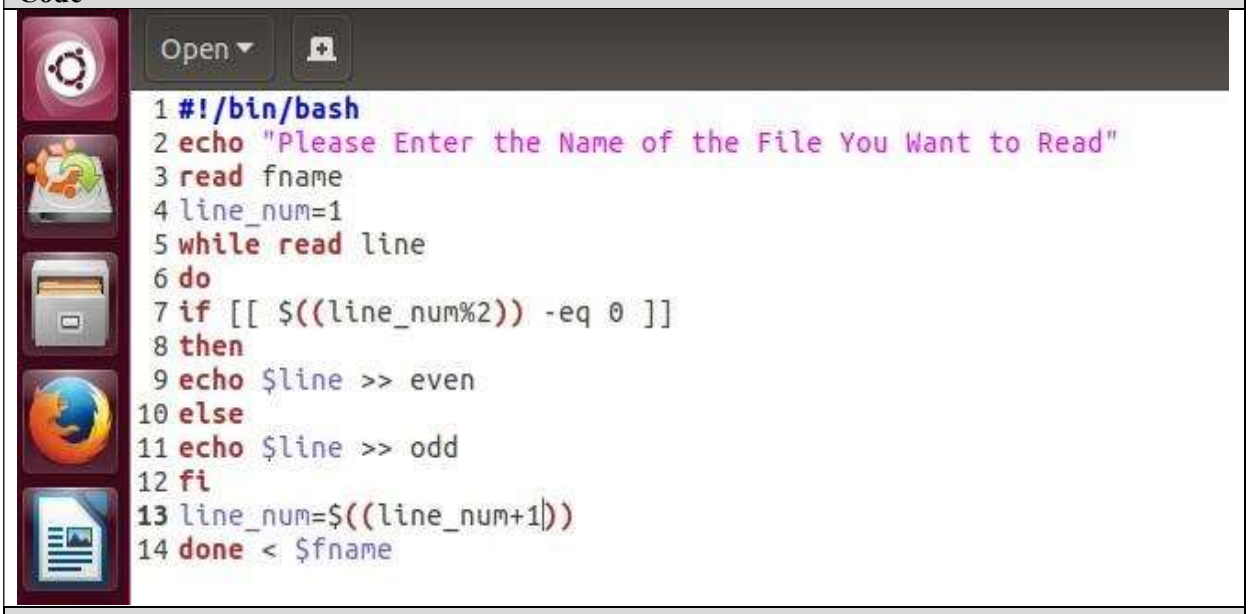
[ABC][0-9]) echo "is A, B, or C followed by a digit." ;; ???) echo "is three characters long." ;; *.txt) echo "is a word ending in '.txt'" ;; *) echo "is something else." ;; esac	**4** **5**

Activity 3:

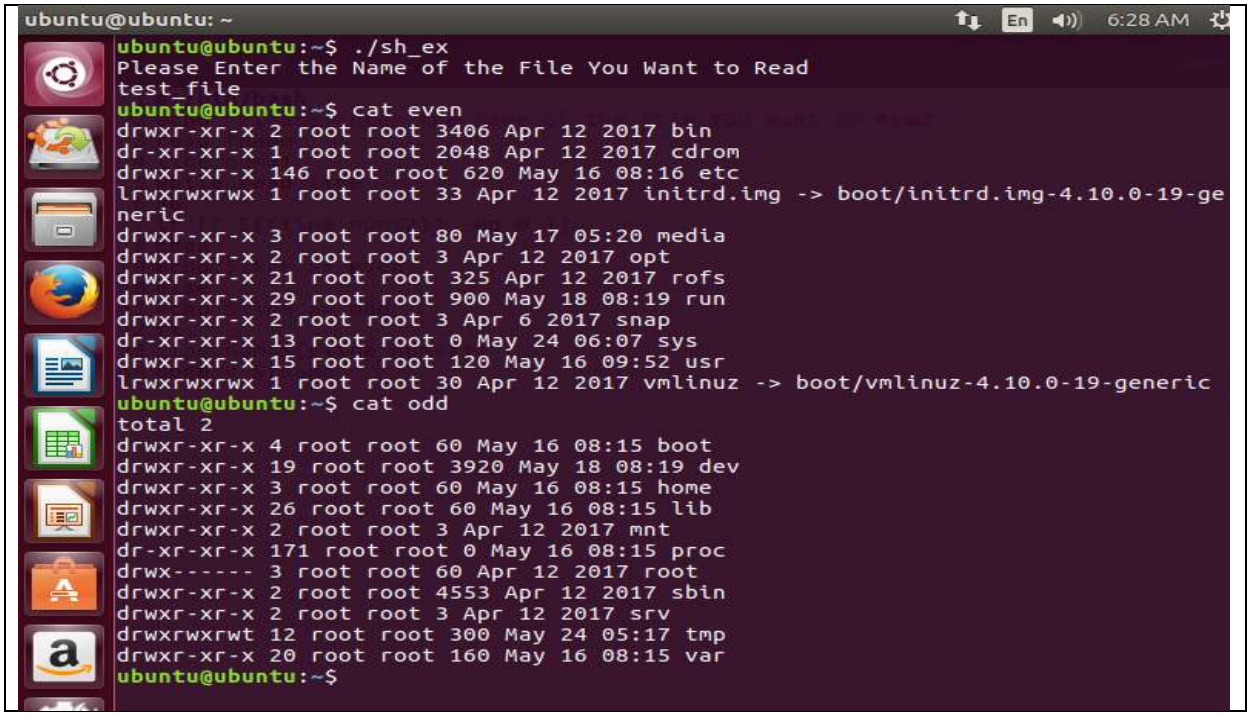
Write a shell script that, given a filename as the argument will write the even numbered line to a file with name even file and odd numbered lines in a text file called odd file.

Solution:

Code



Output



3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Task 1:

Write a shell script that asks the user to enter the marks for three subjects and calculates the GPA for each subject along with the CGPA.

Task 2:

Write a menu-driven shell script that gives four options A, B, C and Q to the user to select one of them. If user enters A then it displays the host-name and uptime, if user enters B then it gives information about disk and memory space, if user enter C then it gives information about home space utilization and if user enters Q then it quits the program.

Task 3:

Write a shell script that, given a filename as the argument will count vowels, blank spaces, characters, number of line and symbols.