UNIX Programming Laboratory

[As per Choice Based Credit System (CBCS) scheme]

SEMESTER – I

Subject Code	20MCA108L	CIE Marks	50
Number of Lecture Hours/Week	02 Hrs Laboratory	SEE Marks	50
		SEE Hours	03

CREDITS – 1

Course Outcome (CO): At the end of this course, the students will be able to:

CO1: Understand the Unix programming environment.

CO2: Be fluent in the use of Vi editor.

CO3: Be able to design and implement shell scripts to manage users with different types of permission and file based applications.

CO4: Be fluent to write shell scripts.

CO5: Evaluate different commands with sample shell scripts

Laboratory Experiments:

Explore the Unix environment and Explore vi editor with vim tutor. Perform the following operations using vi editor, but not limited to:

1. Insert character, delete character, replace character

2. save the file and continue working

3. save the file and exit the editor

4. quit the editor

5. quit without saving the file

6. rename a file

7. insert lines, delete lines,

8. setline numbers

9. search for a pattern

10. move forward and backward

1. Develop a shell script that takes a valid directory name as an argument and recursively descend all the subdirectories, finds the maximum length of any file in that hierarchy and writes this maximum value to the standard output.

2. Develop shell script to implement terminal locking (similar to the lock command). It should prompt the user for a password. After accepting the password entered by the user, it must prompt again for the matching password as confirmation and if match occurs, it must lock the keyword until a matching password is entered again by the user, Note that the script must be written to disregard BREAK, control-D. No time limit need be implemented for the lock duration.

3. Develop a shell script that displays all the links to a file specified as the first argument to the script.The second argument, which is optional, can be used to specify in which the search is to begin.If this second argument is not present, the search is to begin in current working directory. In either case, the starting directory as well as all its sub directories at all levels must be searched. The script need not include any error checking.

4. Write a shell script that accept one or more file names as argument and convert all of them to uppercase, provided they exists in current directory.

5. Implement a shell script to list all the files in a directory whose filename is at least 10 characters. (us expr command to check the length)

6. Develop a shell script that accept a list of filenames as its argument, count and report occurrence of each word that is present in the first argument file on other Argument files.

7. Develop a shell script that reports the logging in of a specified user within one minute after he/she login. The script automatically terminate if specified user does not login during a specified period of time.

8. Develop a shell script that folds long lines into 40 columns. Thus any line that exceeds 40 characters must be broken after 40th, a “\” is to be appended as the indication of folding and the processing is to be continued with the residue. The input is to be supplied through a text file created by the user.

9. Write a shell script that accepts the filename, starting and ending line number as an argument and display all the lines between the given line number.

10. Write a shell script that accepts two file names as arguments, checks if the permissions for these files are identical and if the permissions are identical, output common permissions and otherwise output each file name followed by its permissions

Lab1.sh: Develop a shell script that takes a valid directory name as an argument and recursively descend all the subdirectories, finds the maximum length of any file in that hierarchy and writes this maximum value to the standard output.

```
clear
 dir=$1
   if [ -d $dir ]
   then
     ls -lR $dir | tee f1
     cut –d “ “ –f5  f1 > f2
     sort -n f2 > f3
    echo "Maximum file length is "
    cat f3 | tail -1
   else
    echo "The $dir is not a directory"
fi
```

Run:$sh Lab1.sh one (“one” is a directory should contain one or more files)

Output: one:

```
total 12
-rw-r--r--  1 root root 253 Dec 26 14:03 2a.sh
-rw-r--r--  1 root root 191 Dec 26 14:03 4a.sh
-rw-r--r--  1 root root 389 Dec 26 14:03 7a.sh
Maximum file length is
389
```

Lab2.sh: Develop shell script to implement terminal locking (similar to the lock command). It should prompt the user for a password. After accepting the password entered by the user, it must prompt again for the matching password as confirmation and if match occurs, it must lock the keyword until a matching password is entered again by the user, Note that the script must be written to disregard BREAK, control-D. No time limit need be implemented for the lock duration.

```
stty -echo
echo -e "Enter the password:\c"
read k1

        if [ -z $k1 ]
        then
                echo -e "Invalid Password"
                stty echo
                exit
        else
                echo -e "\n retype password:\c"
                read k2
           if [ $k1 = $k2 ]
           then
            tput clear
      echo -e "\n\n\n\t\t\t****** Terminal Locked****** \n"

              until [ "$k3" = "$k2" ]
              do
                  read k3
              done
           else
               echo -e "\n\n Incorrect Password"
  fi
        fi
    stty echo
```

....

Run: $sh Lab2.sh

```
Enter the password:(whatever u type its invisible)
 retype password:(whatever u type its invisible)
           ****** Terminal Locked******
```

(To unlock the terminal once again type the same password which u typed earlier to get into the shell prompt ‘$’. Password here it is also invisible.)

Lab3.sh : Develop a shell script that displays all the links to a file specified as the first argument to the script. The second argument, which is optional, can be used to specify in which the search is to begin. If this second argument is not present, the search is to begin in current working directory. In either case, the starting directory as well as all its subdirectories at all levels must be searched. The script need not include any error checking.

```
clear
file=$1
        if [ $# -eq 1 ]
        then
                dirx="."
        set - `ls -l $file`
        link=$2
        if [ $link -eq 1 ]
        then
                echo "no other links"
        else
        set - `ls -i $file`
        inode=$1
        find "$dirx" -xdev -inum $inode -print
        fi
              echo "no of links = $link"
fi
```

....

Run: $sh Lab3.sh file1

Output: no other links
no of links = 1

Run: $ln file1 file8

Run: $sh 5a.sh file1

Output:

```
./file8
./file1
no of links = 2
```

Run: $ mkdir dir

Run: $ ln file1 dir/file2

Output: ./file8

```
./file1
./dir/file2
no of links = 3
```

Lab4.sh: Write a shell script that accept one or more filenames as argument and convert all of them to uppercase, provided they exist in current directory.

```
if [ $# -eq 0 ]
then
echo "no arguments"
else
for file in $*
do
if [ -e $file ]
then
upper=`echo $file | tr '[a-z]' '[A-Z]'`
echo "file is converted into :$upper"
else
echo "file does not exist"
fi
done
fi
```

....

Run: $sh Lab4.sh file1 file2

Output:

```
file is converted into :FILE1
file is converted into :FILE2
```

Lab5.sh Implement a shell script to list all the files in a directory whose filename is at least 10 characters. (use expr command to check the length)

```
clear
ls  > file2
for fname in `cat file2`
do
  if [ -f $fname -a `expr "$fname" : '.*'` -gt 10 ] then
   echo  "$fname"
   fi
done
```

Run: $sh Lab5.sh

Output: filenameexpr.sh
Filename10ch.sh

Lab6.sh: Develop a shell script that accept a list of filenames as its argument, count and report occurrence of each word that is present in the first argument file on other argument files.

```
clear
        if [ $# -lt 2 ]
        then
        echo "Enter atleast two filenames as arguments"
        exit
        fi
  for word in `cat $1`
  do
        for file in $*
        do
           if [ "$file" != "$1" ]
           then
       echo "the word frequency of --$word--in the file is:
                `grep -iow $word $file | wc -w`"
           fi
        done
  done
```

....

Create: $cat > file5

```
jan
feb
mar
apr
may
sun
mon
```

Create: cat > file6

```
Mon
tue
wed
jan
apr
thu
fri
```

Run: $sh Lab6.sh file5 file6

Output:

```
the word frequency of --jan--in the file is:
              1
the word frequency of --feb--in the file is:
              0
the word frequency of --mar--in the file is:
              0
the word frequency of --apr--in the file is:
              1
the word frequency of --may--in the file is:
              0
the word frequency of --sun--in the file is:
              0
the word frequency of --mon--in the file is:
              1
```

Lab7.sh: Develop a shell script that reports the logging in of a specified user within one minute after he/she log in. The script automatically terminate if specified user does not log in during a specified period of time.

```
clear
echo -n "enter the login name of the use:"
read login
period=0
echo -n "enter the unit of time (min):"
read min
        until who | grep -w "$login" >/dev/null
        do
                sleep 60
                period=`expr $period + 1`
          if [ $period -gt $min ]
          then
   echo "User:$login has not logged in since $min minutes."
          exit
          fi
        done
echo "User:$login has now logged in."
```

....

Run:$sh Lab7.sh

Input:

```
enter the login name of the use:root
enter the unit of time (min):1
```

Output:

```
User:root has now logged in.
```

Run: $sh Lab7.sh

Input:

```
enter the login name of the use:vijay
enter the unit of time (min):1
(wait for 60 seconds)
```

Output:

```
User:vijay has not logged in since 1 minutes.
```

Lab8.sh: Develop a shell script that folds long lines into 40 columns. Thus any line that exceeds 40 characters must be broken after 40th, a “ “ is to be appended as the indication of folding and the processing is to be continued with the residue. The input is to be supplied through a text file created by the user.

```
clear
echo "Enter the file name"
read file
width=40
line=`cat $file`
echo $line | fold -w "$width" > textfile
sed 's/\(.\{40\}\)/\1\\/' textfile
exit 0
```

....

create: cat > file6

```
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

Run: $sh Lab8.sh

Input:

```
Enter the file name
file6
```

Output:

```
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb\
bbbbbbbbbbbb
```

Lab9.sh: Write a shell script that accept the file name, starting and ending line number as an argument and display all the lines between the given line number

```
clear
if [ $# -ne 3 ]
then
        echo "Pass minimum three argument"
exit
fi
    c=`cat $1 | wc -l`

  if [ $2 -le 0 -o $3 -le 0 -o $2 -gt $3 -o $3 -gt $c ]
  then
      echo "Invalid Input"
  exit
  fi

   sed -n "$2, $3p" $1
```

....

create: cat > filen

```
            jan
            feb
            mar
            apr
            may
            jun
            july
            aug
            sept
```

Run:$sh Lab9.sh filen 3 6

Output:

```
            mar
            apr
            may
            june
```

Lab10.sh: Write a shell script that accepts two file names as arguments, checks if the Permissions for these files are identical and if the permissions are identical, output common permissions and otherwise output each file name followed by its permissions.

```
 clear
f1=$1
f2=$2
if [ -e $f1 -a -e $f2 ]
then
        per1=`ls -l $f1 | cut -d" " -f1`
        per2=`ls -l $f2 | cut -d" " -f1`
if [ $per1 = $per2 ]
then
        echo "Permissions are equal"
        echo "$f1=$per1"
        echo "$f2=$per2"
else
        echo "Permissions are not equal"
        echo "$f1=$per1"
        echo "$f2=$per2"
fi
else
        echo "File does not exist"
fi
```

....

Create two files: $cat > file1 and $cat > file2

Run: $sh Lab10.sh file1 file2

Output:

```
Permissions are equal
          file1=-rw-r--r--
          file2=-rw-r--r--
```

Change the permission for file1: $chmod +x file1 [Enter]

Run:$sh Lab10.sh file1 file2

Output:

```
Permissions are not equal
            file1=-rwxr-xr-x
            file2=-rw-r—r—
```