Sri Siddhartha Institute of Technology, Tumakuru

(A Constituent college of Sri Siddhartha Academy of Higher Education)

INFORMATION SCIENCE AND ENGINEERING

UNIX and SHELL Programming

LAB MANUAL (IS3LB1)

(III SEMESTER)

(2022-2023)

PREPARED BY

Mr.PRADEEP M
Asst. Professor
Dept. of ISE.,

Mrs.Varamahalakshmi M
Asst. Professor
Dept. of ISE.,

Syllabus for the Academic Year – 2021 - 2022

Department: Information Science and Engineering

Semester: 3

Subject Name: UNIX and SHELL Programming Laboratory

Subject Code: IS3LB1 **L-P-C: 0-3-1.5**

Course Objectives:

Sl. No.	Description
1	**Provide introduction to UNIX Operating System and its File System**
2	**Gain an understanding of important aspects related to the SHELL and the process**
3	**Develop the ability to formulate regular expressions and use them for pattern matching.**
4	**Provide a comprehensive introduction to SHELL programming, services and utilities.**

1. **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, outputs the common permissions, otherwise outputs each file name followed by its permissions.**

```
echo "Enter the First File Name:"
read f1
echo "Enter the Second File Name:"
read f2
if [ ! -f $f1 ] || [ ! -f $f2 ]
then
    echo "Given File Name is not Exited"
    exit 0 ;
fi
ls -l $f1 2> /dev/null > t1
x=`cut -c 2-10 t1`
ls -l $f2 2> /dev/null > t2
y=`cut -c 2-10 t2`
echo "Permission of the First File is $x"
echo "Permission of the Second File is $y"
if [ $x = $y ]
then
     echo "Permission for both files are same"
     echo $x
else
      echo "Permissions are different"
      echo $x $f1
      echo $y $f2
```

fi

Script name : 1a.sh
// create three files : two with same permissions and one with different permissions .
Input/Output :
// files with different permissions
[root@localhost ~]$ sh 1a.sh
Enter the First File Name:
1a.sh
Enter the Second File Name:
1b.c
Permission of the First File is rwxrwxrwx
Permission of the Second File is rw-r--r--
Permissions are different
rwxrwxrwx 1a.sh
rw-r--r-- 1b.c

// files with same permissions
[root@localhost ~]$ sh 1a.sh
Enter the First File Name:
sample1
Enter the Second File Name:
sample2
Permission of the First File is rw-r--r--
Permission of the Second File is rw-r--r--
Permission for both files are same
rw-r—r--

// files which donot exist
[root@localhost ~]$ sh 1a.sh
Enter the First File Name:
1
Enter the Second File Name:
2
Given File Name is not Exited

2. **Write a Shell script that takes a valid directory name as an argument and recursively descends all the subdirectories, finds the maximum length of any file in that hierarchy and writes this maximum value to the standard output.**

```
echo "ENTER THE NAME OF THE DIRECTORY:"
read dir;
ls -lR $dir | cut -c 24-28 | sort -n | tail -1
```

Input/Output :

```
[root@localhost ~]$ sh 2a.sh
ENTER THE NAME OF THE DIRECTORY:
unix
4096
```

3. **Write a Shell script that accepts valid log-in names as arguments and prints their corresponding home directories. If no arguments are specified, print a suitable error message.**

```
y=$#
i=1
if [ $y -eq 0 ]
then
        echo "MAKE SURE TO ENTER THE ARGUMENTS!!"
else

    while [ $i -le $y ]
    do
      loginname=$1
     grep $loginname /etc/passwd>s
     if [ $? -eq 0 ]
     then
        echo "LOGINNAME: $loginname"
        echo "HOME DIRECTORY"
        cut -d \: -f 6 s
     else
         echo "ERROR--LOGINNAME IS ABSENT"
     fi
   shift
    i=`expr $i+1`
    done
fi
```

Input/Output :

```
[root@localhost ~]$ sh 3a.sh ise060
LOGINNAME: ise060
HOME DIRECTORY
/home/ise060
```

4. **Write a Shell script to that accepts path names and creates all the components in that path name as directories. For example, if the script name is mpe, then the command mpe a/b/c/d should create directories a, a/b, a/b/c and a/b/c/d.**

```
echo "ENTER THE PATHNAME:"
read p
j=1
i=1
len=`echo $p | wc -c`
while [ $i -le $len ]
do
  x=`echo $p | cut -d \/ -f $j`
  namelength=`echo $x | wc -c`
  mkdir -v $x
  cd $x
  j=`expr $j + 1`
```

```
  i=`expr $i + $namelength`
done
```

Input/Output :

```
[root@localhost ~]$ sh 4a.sh
ENTER THE PATHNAME:
ab/bc/ef
mkdir: created directory `ab'
mkdir: created directory `bc'
mkdir: created directory `ef'
```

5. **Write a Shell script to find and display 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 the directory 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.**

```
if [ $# -eq 0 ];
then
        echo "no arguments"
        exit
fi
if [ $# -eq 2 ];then
        dir=$2
else
        dir=`pwd`
fi
inode=`stat -c %i $1`
count=0
for link in `find $dir -inum $inode`
do
        echo $link
        count=`expr $count + 1`
done
if [ $count -eq 0 ];
then
        echo "$1 has no link in the directory $dir"
else
        echo "$1 has $count links in the directory $dir"
fi
```

Input/Output :

```
[root@localhost ~]# sh 5a.sh 7b.c
/root/7b.c
/root/9
/root/10
/root/11
/root/12
7b.c has 5 links in the directory /root
```

6. **Write a Shell script that reports the logging in of a specified user within one minute after he/she log in. The script automatically terminated if specified user does not log in during a specified period of time.**

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

Input /output:

```
1.Enter the login name of the user:
root
Enter the unit of time(min):
1
root has now logged in.


2.Enter the login name of the user:
iseexam
Enter the unit of time(min):
1
iseexam has not logged in since 1 minutes
```

7. **Write a Shell script that determine the period for which a specified user is working on system.**

```
echo  "enter the user name :\c"
read usr
tuser=`who | tr -s " " | head -1 | cut -d " " -f1`
if [ "$tuser" = "$usr" ]
then
     tm=`who | tr -s " " | head -1 | cut -d " " -f4`
     uhr=`echo $tm | cut -d ":" -f1`
     umin=`echo $tm | cut -d ":" -f2`
     shr=`date "+%H"`
     smin=`date "+%M"`
```

```
if [ $smin -lt $umin ]
then
     shr=`expr $shr - 1`
     smin=`expr $smin + 60`
fi
h=`expr $shr - $uhr`
m=`expr $smin - $umin`
echo "user name : $usr"
echo "login period : $h : $m"
else
     echo "Invalid User"
fi
```

Input /output:

```
[root@localhost ~]# sh 7a.sh

1.enter the user name :ise070

user name: ise070

login period: 1:10

2. enter the user name : iseexam

Invalid User
```

8. **Write 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.**

```
if [ $# -lt 2 ]
then
      echo "USAGE:sh $0<pattern file><file1><file2>------"
      exit 0;
fi
file=`cat $1`
shift
for word in $file
do
    for file in $*
    do
      echo "occerence count of $word in file $file is:"
      echo "`grep -iow $word $file | wc`"
    done
done
```

Input /output:

contents of file which are given as arguments

patternfile.txt

hi

how are you

Samplefile.txt

hi hi hi hi hi

how how how how how

are are are

you

```
[root@localhost ~]# sh 8a.sh patternfile.txt samplefile.txt
occerence count of hi in file he is:
    5      5      15
occerence count of how in file he is:
    5      5      20
occerence count of are in file he is:
    3      3      12
occerence count of you in file he is:
    1      1       4
```

9. **Write a Shell script that deletes all lines containing a specific word in one or more file supplied as argument to it.**

```
clear
if [ $# -lt 1 ]
then
echo " GIVE THE INPUT STRING"
exit
fi
echo "INPUT THE WORD TO DELETE"
read word
for file in $*
do
grep -iv "$word" $file|tee /dev/null $file
done
echo done
```

Input /output:

samplefile.txt

hi hi hi hi hi

how how how how how

are are are

you

1.[root@localhost ~]# sh 9a.sh

GIVE THE INPUT STRING

2.[root@localhost ~]# sh 12a.sh samplefile.txt
INPUT THE WORD TO DELETE
hi
how how how how how
are are are
you

done

10. **Write a Shell script that accepts file names specified as arguments and creates a shell script that contains this file as well as the code to recreate these files. Thus if the script generated by your script is executed, it would recreate the original files.**

```
for i in $*
do
  echo "echo $i file is recreated"
   echo "cat > $i << *"
   cat $i
  echo "*"
done
```

Input/Output :

```
[root@localhost ~]# sh 10a.sh 3a.sh
echo 3a.sh file is recreated
cat > 3a.sh << *
echo "ENTER THE NAME OF THE DIRECTORY:"
read dir;
ls -lR $dir | cut -c 24-28 | sort -n | tail -1
*
```

Course Outcomes:

Sl. No.	Description
1	Describe the architecture and features of UNIX Operating System and distinguish it from other Operating System
2	Demonstrate UNIX commands for file handling and process control.
3	Demonstrate Regular expressions for pattern matching and apply them to various filters for a specific task.
4	Analyze a given problem and apply requisite facets of SHELL programming in order to devise a SHELL script to solve the problem.