AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

LABORATORY MANUAL

OPERATING SYSTEM

SUBJECT CODE: 3140702

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT B.E. 4th SEMESTER

NAME: _____

ENROLLMENT NO: _____

BATCH NO: _____

YEAR: _____

Amiraj College of Engineering and Technology,
Nr.Tata Nano Plant, Khoraj, Sanand, Ahmedabad.

AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

Amiraj College of Engineering and Technology,

Nr.Tata Nano Plant, Khoraj, Sanand, Ahmedabad.

CERTIFICATE

This is to certify that Mr. / Ms. _____

Of class_____ Enrolment No _____has

Satisfactorily completed the course in _____as

by the Gujarat Technological University for ____ Year (B.E.) semester___ of

Computer Science and Engineering in the Academic year _____.

Date of Submission:-

Faculty Name and Signature

Foram Patel

Head of Department

(CSE)

AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

B.E. 4th SEMESTER

SUBJECT: OPERATING SYSTEM

SUBJECT CODE: 3140702

List of Experiments

Sr. No.	Date	Page No.	Experiments	Teacher's Sign.	Marks
1			Study of Basic commands of Linux/UNIX.		
2			Write a shell script to generate marksheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.		
3			Write a shell script to find factorial of given number n.		
4			Write a shell script which will generate first n fibonnacci numbers like: 1, 1, 2, 3, 5		
5			Write a menu driven shell script which will print the following menu and execute the given task. a. Display calendar of current month b. Display today's date and time c. Display usernames those are currently logged in the system d. Display your name at given x, y position e. Display your terminal number		
6			Write a shell script to check entered string is palindrome or not.		
7			Write a shell script to validate the entered date.		
8			Write a shell script to convert each word in a given text into capital.		
9			Study of Unix Shell and Environment Variables.		
10			Study of Advance commands and filters of Linux/UNIX.		

Faculty Member
Foram Patel

Department coordinator
Foram Patel

Practical -1

Aim: Explain and run basic commands- (cal,cat,cc,cd,chdir,clear,cls,cmp,copy,cp,date,ed,edit,exit,find,ls,man,mkdir,mv,printf, ps, pwd, rm,rmdir,sleep,wc).

commands:

1) Cal: Displays a calendar

Syntax: - cal [options] [month] [year]

Description:-

cal displays a simple calendar. If arguments are not specified, the current month is displayed. The switching options are as follows:

-1	Display single (current) month output. (This is the default.)
-3	Display prev/current/next month output
-s	Display Sunday as the first day of the week (This is the default.)
-m	Display Monday as the first day of the week
-j	Display Julian dates (days one-based, numbered from January 1)
-y	Display a calendar for the current year

2) cat: It is used to create, display and concatenate file contents.

Syntax: - cat [options] [FILE]...

Description:-

-A	Show all.
-b	Omits line numbers for blank space in the output.
-e	A $ character will be printed at the end of each line prior to a new line.
-E	Displays a $ (dollar sign) at the end of each line.

-n	Line numbers for all the output lines.
-s	If the output has multiple empty lines it replaces it with one empty line.
-T	Displays the tab characters in the output.
-v	Non-printing characters (with the exception of tabs, new-lines and form-feeds) are printed visibly.

Two basically three uses of the cat command.

1) Create new files.
2) Display the contents of an existing file.
3) Concatenate the content of multiple files and display.

3) cd: It is used to change the directory.

Syntax: - cd [directory]

Description:-

Used to go back one directory on the majority of all UNIX shells. It is important that the space be between the cd and directory name.

4) cp: - cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

Syntax: - cp [options]... source destination

Description:-

- Here, after cp command contents of both source file and destination file files are the same.
- It will copy the content of source file to destination file.
- If the destination file doesn't exist, it will be created.
- If it exists then it will be overwritten without any warning.
- If there is only one file to be copied then destination can be the ordinary file or the directory file.

-a	archive files
-f	force copy by removing the destination file if needed
-i	interactive - ask before overwrite
-l	link files instead of copy
-L	follow symbolic links
-n	no file overwrite
-u	update - copy when source is newer than dest

5) clear: - It clears the terminal screen.

Syntax: - clear

Description:-

- Clear clears your screen if this is possible, including its scroll back buffer.

- Clear ignores any command-line parameters that may be present.

.

6) mkdir:- This command is used to create a new directory

Syntax: - mkdir [options] directory

Description:-

-m	Set permission mode (as in chmod)
-p	No error if existing, make parent directories as needed.
-v	Print a message for each created directory
directory	The name of the directory that you wish to create

7) cmp:- It compares two files and tells you which line numbers are different.

Syntax : - cmp [options..] file1 file2

Description:-

Let's create a file named os2. And use cmp command to compare os and os1files.

-c	Output differing bytes as characters.
-l	Print the byte number (decimal) and the differing byte values (octal) for each difference.
-s	Prints nothing for differing files, return exit status only.
-c	Output differing bytes as characters.

8) cp:- cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

Syntax: - cp [options]... source destination

Description:-

- Here, after cp command contents of both source file and destination file files are the same.
- It will copy the content of source file to destination file.
- If the destination file doesn't exist, it will be created.
- If it exists then it will be overwritten without any warning.
- If there is only one file to be copied then destination can be the ordinary file or the directory file.

-a	archive files
-f	force copy by removing the destination file if needed
-i	interactive - ask before overwrite
-l	link files instead of copy
-L	follow symbolic links
-n	no file overwrite

-u	update - copy when source is newer than dest

9) bc:- bc command is used for command line calculator. It is similar to basic calculator. By using which we can do basic mathematical calculations.

Syntax:bc[options]

Description:-

- bc is a language that supports arbitrary precision numbers with interactive execution of statements.
- bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read.

-q	To avoid bc welcome message
-l	To include math library functionalities

10) exit:Exit immediately after writing the last line of the last file in the argument list.

11) find:- Finds one or more files assuming that you know their approximate path.

Syntax :- find [options] path

Description :-

Find is one of the powerful utility of Unix (or Linux) used for searching the files in a directory hierarchy

path	A path name of a starting point in the directory hierarchy
-maxdepth	Descend at most levels (a non-negative integer) levels of directories below the command line arguments.
-i	ignore the case in the current directory and sub-directories.
-size	Find file based on size

12) ls:- Lists the contents of a directory

Syntax :- ls [options]

Description :-

-a	Shows you all files, even files that are hidden (these files begin with a dot.)
-A	List all files including the hidden files. However, does not display the working directory (.) or the parent directory (..).
-d	If an argument is a directory it only lists its name not its contents
-l	Shows you huge amounts of information (permissions, owners, size, and when last modified.)
-	Displays a slash (/) in front of all directories

p	
-r	Reverses the order of how the files are displayed
-R	Includes the contents of subdirectories

13) man:- man command which is short for manual, provides in depth information about the requested command (or) allows users to search for commands related to a particular keyword.

Syntax:- man commandname [options]

Description :-

-a	Print a one-line help message and exit.
-k	Searches for keywords in all of the manuals available.

14) mkdir:- This command is used to create a new directory

Syntax :- mkdir [options] directory

Description :-

-m	Set permission mode (as in chmod)
-p	No error if existing, make parent directories as needed.
-v	Print a message for each created directory
directory	The name of the directory that you wish to create

15) mv:- It is used to move/rename file from one directory to another.

Syntax :- mv [options] oldname newname

Description :-

- mv command which is short for move.
- mv command is different from cp command as it completely removes the file from the source and moves to the directory specified, where cp command just copies the content from one file to another.
- mv has two functions: it renames a file and it moves a group of files to a different directory. Mv doesn't create a copy of the file , it merely renames it. No additional space is consumed on disk during renaming. For example if we rename a file os to os1 and then if we try to read file os we will get error message as it is renamed to os1 there is no existence of file named os.

16) ps:- It is used to report the process status. ps is the short name for Process Status.

Syntax:- ps [options]

Description :-

-a	List information about all processes most frequently requested: all those except process group leaders and processes not associated with a terminal

-A	List information for all processes. Identical to -e, below
-f	Generate a full listing
-j	Print session ID and process group ID
-l	Generate a long listing

17) pwd:-Displaying your current directory name (Print working directory).

Syntax:-pwd [options]

Description:-

At the time of logging in user is placed in the specific directory of the file system. You can move around from one directory to another, but any point of time, you are located in only one directory. This directory is known as your current directory.

pwd command tells your current directory.

18) rmdir:- It is used to delete/remove a directory and its subdirectories.

Syntax :- rmdir [options..] Directory

Description :-

It removes only empty directory.

-p	Allow users to remove the directory and its parent directories which become empty.

19) rm:- It is used to remove/delete the file from the directory.

Syntax :- rm [options..] [file|directory]

Description :-

- Files can be deleted with rm. It can delete more than one file with a single invocation. For deleting a single file we have to use rm command with filename to be deleted.
- Deleted file can't be recovered. rm can't delete the directories. If we want to remove all the files from the particular directory we can use the * symbol.

-f	Ignore nonexistent files, and never prompt before removing.
-i	Prompt before every removal.

Example :-

$ rm myfile.txt

- Remove the file myfile.txt. If the file is write-protected, you will be prompted to confirm that you really want to delete it.

$ rm *

- Remove all files in the working directory. If it is write-protected, you will be prompted before rm removes it.

$ rm -f myfile.txt

- Remove the file myfile.txt. You will not be prompted, even if the file is writeprotected; if rm can delete the file, it will.

$ rm -f *

- Remove all files in the working directory. rm will not prompt you for any reason before deleting them.

  $ rm -i *
- Attempt to remove every file in the working directory, but prompt before each file to confirm.

20) sleep:- Delay for a specified amount of time

Syntax :- sleep NUMBER[SUFFIX]

Description:-

- The sleep command pauses for an amount of time defined by NUMBER.

SUFFIX may be "s" for seconds (the default), "m" for minutes, "h" for hours, or "d" for days.14.ed:

21) wc:- Word Count (wc) command counts and displays the number of lines, words, character and number of bytes enclosed in a file.

Syntax: - wc [*options*] [filename]

Description:-

This command counts lines, words and characters depending on the options used. It takes one or more filenames as its arguments and displays four-columnar output. For example let's read our os1 file. And we use wc command with that filename.

-l	print the newline counts.
-w	print the word counts.
-c	print the byte counts.
-m	print the character counts.
-L	print the length of the longest line.

22) finger:- finger command displays the user's login name, real name, terminal name and write status (as a "*" after the terminal name if write permission is denied), idle time, login time, office location and office phone number.

Syntax:- finger [username]

Description :-

-l	Force long output format
-s	Force short output format

PRACTICAL – 2

AIM : Write a shell script to generate marksheet of a student. Take 3 subjects, calculate and display total marks, percentage and Class obtained by the student.

```
echo " Enter name "
read name
echo " Enter enrollment number "
read no
echo " Enter your marks "
read m1
read m2
read m3
total=$(expr $m1 + $m2 + $m3 )
avg=$(expr $total / 3)
                    echo "Student Name : $name"
                    echo "EnrollMent NUmber: $no"
                    echo "Average is : $avg"
if [ $m1 -ge 35 ] && [ $m2 -ge 35 ] && [ $m3 -ge 35 ]
        then
                    echo "Result is: Pass"
if [ $avg -ge 80 ] && [ $avg -le 100 ]
        then
                    echo "Result is: Distinction"
 elif [ $avg -ge 61 ] && [ $avg -le 79 ]
        then
        echo "Result is: First class"
 elif [ $avg -ge 35 ] && [ $avg -le 60 ]
        then
        echo "Result is: Second class"
fi
        else
                    echo "Result is: fail"
fi
```

O/P :

```
Enter name
DJ
 Enter enrollment number
13
 Enter your marks
89
90
98
Average is : 92
Result is: Distinction
```

PRACTICAL – 3

AIM : Write a shell script to find factorial of given number n.

```
echo "Enter the number to find factorial?"
read number
fact=1
while [ $number -gt 0 ]
do
fact=`expr $number \* $fact`
number=`expr $number - 1`
done
echo "factorial is : $fact"
```

O/P:

```
DJ@ubuntu:~/Downloads/Practicals/OS _Practicals$ bash fab.sh
Enter a number

3

6

Enter a number

4

24
```

PRACTICAL – 4

AIM : Write a shell script which will generate first n fibonnacci numbers like: 1, 1, 2, 3, 5

```
N=6
a=0
b=1

echo "The Fibonacci series is : "

for (( i=0; i<N; i++ ))
do
   echo -n "$a "
   fn=$((a + b))
   a=$b
   b=$fn
done
# End of for loop
```

Output:

```
Fibonacci Series is:
0
1
1
2
3
5
8
```

PRACTICAL – 5

AIM : Write a menu driven shell script which will print the following menu and execute the given task.
a. Display calendar of current month
b. Display today's date and time
c. Display usernames those are currently logged in the system
d. Display your name at given x, y position
e. Display your terminal number.

```
echo " MENU
a.    . Display calendar of current month
b.    . Display today's date and time
c.    . Display usernames those are currently logged in the system
d.    . Display your terminal number
e.    . Exit

Read i
Case "$i" in
1)    cal ;;
2)    date ;;
3)    who;;
4)    tty ;;
5)    exit ;;
*) echo "enter valid in put" ;;
esac
```

Output:

```
1) DJ@ubuntu:~/Downloads/Practicals/OS _Practicals$ cal
   February 2018
Su Mo Tu We Th Fr Sa
              1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28

2) DJ@ubuntu:~/Downloads/Practicals/OS _Practicals$ date
Mon Feb 26 20:16:47 PST 2018

3) DJ@ubuntu:~/Downloads/Practicals/OS _Practicals$ who
paras    tty7         2018-02-26 19:48 (:0)

4) DJ@ubuntu:~/Downloads/Practicals/OS _Practicals$ tty
/dev/pts/1
```

PRACTICAL – 6

AIM : Write a shell script to check entered number is palindrome or not.

```
num=545

# Storing the remainder
s=0

# Store number in reverse
rev=""
temp=$num
 while [ $num -gt 0 ]
do
   # Get Remainder
   s=$(( $num % 10 ))
   # Get next digit
   num=$(( $num / 10 ))

   # Store previous number and
   # current digit in reverse
   rev=$( echo ${rev}${s} )
done
  if [ $temp -eq $rev ];
then
   echo "Number is palindrome"
else
   echo "Number is NOT palindrome"
fi
```

Output:

Number is palindrome

PRACTICAL – 7

AIM : Write a shell script which will accept a number b and display first n prime numbers as output.

```
prime_1=0
echo "enter the range"
read n
echo " Primenumber between 1 to $n is:"
echo "1"
echo "2"
for((i=3;i<=n;))
do
for((j=i-1;j>=2;))
do
if [  `expr $i % $j` -ne 0 ] ; then
prime_1=1
else
prime_1=0
break
fi
j=`expr $j - 1`
done
if [ $prime_1 -eq 1 ] ; then
echo $i
fi
i=`expr $i + 1`
done
```

Output:

```
Enter the range
25
Prime number 1 to 25 is :
1
2
3
5
7
11
13
17
19
23
```

PRACTICAL – 8

AIM : Write a shell script to display multiplication table of given number.

```
clear
echo ----------------------------------
echo '\tMultiplication Table'
echo ----------------------------------
echo Enter table number
read tn
echo Enter how many rows
read n
i=1
while [ $i -le $n ]
do
	k=$(expr $i \* $tn)
	echo "$i * $tn = $k"
	i=$(expr $i + 1)
done
```

Output:

```
Enter table number
6
Enter how many rows
5
1 * 6 = 6
2 * 6 = 12
3 * 6 = 18
4 * 6 = 24
5 * 6 = 30
```

PRACTICAL – 9

AIM : Study of Unix Shell and Environment Variables.

What is an environment variable?
Environment variables or ENVs basically define behavior of the environment. They can affect the processes ongoing or the programs that are executed in the environment.

Scope of an environment variable
Scope of any variable is the region from which it can be accessed or over which it is defined. An environment variable in Linux can have global or local scope.

Global
A globally scoped ENV that is defined in a terminal can be accessed from anywhere in that particular environment which exists in the terminal. That means it can be used in all kind of scripts, programs or processes running in the environment bound by that terminal.

Local
A locally scoped ENV that is defined in a terminal cannot be accessed by any program or process running in the terminal. It can only be accessed by the terminal(in which it was defined) itself.

Variable	Description
USER	The username
HOME	Default path to the user's home directory
EDITOR	Path to the program which edits the content of files
UID	User's unique ID
TERM	Default terminal emulator
SHELL	Shell being used by the user

Practical 10 :

AIM : Study of Advance commands and filters of Linux/UNIX.

Linux Filter commands accept input data from **stdin** (standard input) and produce output on **stdout** (standard output). It transforms plain-text data into a meaningful way and can be used with pipes to perform higher operations.

.

Linux Filter Commands

1. cat

   Syntax:

   cat <**fileName**> | cat or tac | cat or tac |. . .

2. cut

   Syntax:

The 'cut' command is useful in selecting a specific column of a file. After (-d), delimiter (from where you want to separate the columns) comes. Delimiters can be a space (' '), a hyphen (-), a slash (/) or anything else. After (-f), column number is mentioned.

Syntax:

cut -d(delimiter) -f(columnNumber) <**fileName**>

3. grep

The 'grep' command is generally used with pipe **(|)**.

Syntax:

command | grep <**searchWord**>

4. comm

The 'comm' command compares two files or streams. By default, 'comm' will always display **three columns**. First column indicates non-matching items of first file, second column indicates non-matching items of second file, and third column indicates matching items of both the files. Both the files has to be in sorted order for 'comm' command to be executed.

Syntax:

comm <**file1**> <**file2**>

5. sed

Command 'sed' stands for stream editor. You can use this command to edit streams (files) using regular expressions. But this editing is not permanent. It remains only in display, but in actual, file content remains same.

Syntax:

command | sed 's/<**oldWord**>/<**newWord**>/'

6. tee

The 'tee' command is similar to 'cat' command with only one difference. It puts stdin on stdout and also put them into a file.

Syntax:

cat or tac <**fileName**> | tee <**newFile**> | cat or tac |.....

7. tr

The command 'tr' stands for '**translate**'. It is used to translate, like from lowercase to uppercase and vice versa or new lines into spaces.

Syntax:

command | tr <'old'> <'new'>

8. uniq

With the help of uniq command you can form a sorted list in which every word will occur only once.

Syntax:

command <**fileName**> | uniq

9. wc

The 'wc' command helps in counting the lines, words and characters in a file.

Syntax:

1. wc <**fileName**> (Counts words, lines and characters)
2. wc -l <**fileName**> (Counts only lines)
3. wc -w <**fileName**> (Counts only words)
4. wc -c <**fileName**> (Counts only characters)

10. od

The 'od' term stands for octal dump. It displays content of a file in different human-readable formats like hexadecimal, octal and ASCII characters.

Syntax:

1. od -b **<fileName>** (display files in octal format)
2. od -t x1 **<fileName>** (display files in hexadecimal bytes format)
3. od -c **<fileName>** (display files in ASCII (backslashed) character format)