LINUX INTERNALS LABORATORY

LAB MANUAL

Academic Year	**:**	**2019- 2020**
Course Code	**:**	**AIT105**
Regulations	**:**	**IARE - R16**
Class	**:**	**III Year II Semester**
Branch	**:**	**IT**

Department of Information Technology

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal – 500 043, Hyderabad

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

INFORMATION TECHNOLOGY

Program Outcomes	
PO1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
PO3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
PO12	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
Program Specific Outcomes	
PSO1	**Professional Skills:** The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient analysis and design of computer - based systems of varying complexity.
PSO2	**Software Engineering Practices:** The ability to apply standard practices and strategies in software service management using open-ended programming environments with agility to deliver a quality service for business success.
PSO3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

INDEX

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES

Exp. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1	Study and Practice on various commands like man, passwd, tty, script, clear, date, cal, cp, mv, ln, rm, unlink, mkdir, rmdir, du, df, mount, umount, find, unmask, ulimit, ps, who, w.	PO1,PO5	PSO1
2	Study and Practice on various commands like cat, tail, head , sort, nl, uniq, grep, egrep, fgrep, cut, paste, join, tee, pg, comm, cmp, diff, tr, awk, tar, cpio.	PO1,PO5	-
3	e) Write a Shell Program to print all .txt files and .c files. f) Write a Shell program to move a set of files to a specified directory. g) Write a Shell program to display all the users who are currently logged in after a specified time. h) Write a Shell Program to wish the user based on the login time.	PO1,PO5	-
4	e) Write a Shell program to pass a message to a group of members, individual member and all. f) Write a Shell program to count the number of words in a file. g) Write a Shell program to calculate the factorial of a given number. h) Write a Shell program to generate Fibonacci series.	PO1	-
5	a) Simulate cat command b) Simulate cp command	PO1	-
6	a) Simulate tail command b) Simulate head command	PO1,PO3,PO5	-
7	a) Simulate mv command b) Simulate nl command	PO1,PO3,PO5	-
8	Write a program to handle the signals like SIGINT, SIGDFL, SIGIGN	PO1,PO2,PO5	-
9	Implement the following IPC forms a) FIFO b) PIPE	PO1,PO2,PO5	-
10	1. Write a C program (sender.c) to create a message queue with read and write permissions to write 3 messages to it with different priority numbers. 2. Write a C program (receiver.c) that receives the messages (from the above message queue as specified and displays them.	PO1,PO2, PO3, PO5	-
11	Implement of shared memory form of IPC.	PO1,PO2,PO5	-
12	1. Write client and server programs (using c) for interaction between server and client processes using TCP Elementary functions. 2. Write client and server programs (using c) for interaction between server and client processes using UDP Elementary functions.	PO1,PO2,PO5	-

LINUX INTERNALS LABORATORY

OBJECTIVE:

The Linux Internals laboratory course covers major methods of Inter Process Communication (IPC), which is the basis of all client / server applications under Linux, Linux Utilities, working with the Bourne again shell (bash), files, process and signals. There will be extensive programming exercises in shell scripts. It also emphasizes various concepts in multithreaded programming and socket programming.

OUTCOMES:

Upon the completion of Linux Internals practical course, the student will be able to attain the following:

1. Familiar with the Linux Command-line environment
2. Understand system administration process by providing hands-on experience.
3. Understand Process Management and inter process communications techniques.

EXPERIMENT - 1

BASIC COMMANDS I

1.1 OBJECTIVE:

To Study and Practice on various commands like man, passwd, tty, script, clear, date, cal, cp, mv, ln, rm, unlink, mkdir, rmdir, du, df, mount, umount, find, unmask, ulimit, ps, who, w.

1.2 RESOURCES:

Linux operating system, vi-editor, shell-interpreter

1.3 DESCRIPTION / PROCEDURE

1. Open Linux Operating System Command Line Interface.
2. Execute command with options in shell prompt.
3. Press ctrl +z to exit from process.

Command	man (MANUAL)
Syntax / Synopsis	**man** [**-acdfFhkKtwW**] [**--path**] [**-m** *system*] [**-p***string*] [**-C** *config_file*] [**-M** *pathlist*] [**-P** *pager*] [**-S***section_list*] [*section*] *name*
Description	man - format and display the on-line manual pages . If you specify *section*, **man** only looks in that section of the manual. *name* is normally the name of the manual page, which is typically the name of a command, function, or file
Example	**$man ./foo.5** or even **man /cd/foo/bar.1.gz**. $man grep

Command	passwd
Syntax	passwd [-r | files | -r nis | -r nisplus] [-a] [-d | -l] [-e] [-f] [-g] [-h] [-n min] [-s] [-w warn] [-x max] [-D domainname][name]
Description	passwd is a text file, that contains a list of the system's accounts, giving for each account some useful information like user ID, group ID, home directory, shell, etc. Often, it also contains the encrypted passwords for each account.
Example	$passwd Current Password: XXXX New Password: YYYYY Confirm New Password: YYYY

Command	tty
Syntax	tty – [option]
Description	Print the file name of the terminal connected to standard input.
Example	$tty /dev/pts/14

Command	script
Syntax	script [-a] [-f] [-q] [-t] [file]
Description	script makes a typescript of everything printed on your terminal. It is useful for students who need a hardcopy record of an interactive session as proof of an assignment, as the typescript file can be printed out later
Example	$script filename

Command	clear
Syntax	clear
Description	clear clears your screen if this is possible
Example	$clear

Command	date
Syntax	date [OPTION]... [+FORMAT]
Description	Display the current time in the given FORMAT, or set the system date.
Example	*$date "+%m-%d-%Y %B"* *02-10-2010 February*

Command	cal
Syntax	cal [-smjy13] [[month] year]
Description	cal displays a simple calendar. If arguments are not specified, the current month is displayed.
Example	$cal June, 2013 Su Mo Tu We Th Fr Sa 26 27 28 29 30 31 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 29 30 1 2 3 4 5 6

Command	cp
Syntax	cp [OPTION]... --target-directory=DIRECTORY SOURCE...
Description	Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
Example	$cp file1 file2

Command	mv
Syntax	mv [OPTION]... --target-directory=DIRECTORY SOURCE...
Description	Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.
Example	$mv file4 file5

Command	$ln
Syntax	ln [OPTION]... --target-directory=DIRECTORY TARGET...
Description	Create a link to the specified TARGET with optional LINK_NAME. If LINK_NAME is omitted, a link with the same basename as the TARGET is created in the current directory. When using the second form with more than one TARGET, the last argument must be a directory; create links in DIRECTORY to each TARGET
Example	ln -s file5 file6

Command	rm
Syntax	rm [OPTION]... FILE...
Description	This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories
Example	$rm file5

Command	unlink
Syntax	unlink *PATH
Description	unlink deletes a name from the filesystem. If that name was the last link to a file and no processes have the file open the file is deleted and the space it was using is made available for reuse.
Example	$unlink /file2

Command	mkdir
Syntax	mkdir [OPTION] DIRECTORY...
Description	Create the DIRECTORY(ies), if they do not already exist.
Example	$mkdir salary

Command	Rmdir
Syntax	rmdir [OPTION]... DIRECTORY...
Description	Remove the DIRECTORY(ies), if they are empty.
Example	$rmdir salary

Command	du
Syntax	du [OPTION]... [FILE]...
Description	Summarize disk usage of each FILE, recursively for directories.
Example	$du /tmp 4 /tmp/vmware-root 8 /tmp/pulse-xc7xdoM9vB2K . . 8 /tmp/orbit-vivek 4 /tmp/.esd-1000 31644 /tmp

Command	df
Syntax	df [OPTION]... [FILE]...
Description	This manual page documents the GNU version of df. df displays the amount of disk space available on the filesystem containing each file name argument. If no file name is given, the space available on all currently mounted filesystems is shown. (In 1 KB)
Example	$df -h Filesystem Size Used Avail Capacity Mounted on /dev/wd0a 938M 43.0M 848M 5% / /dev/wd0e 817M 2.0K 776M 0% /home /dev/wd0d 2.9G 573M 2.2G 20% /usr

Command	mount
Syntax	mount [-fnrsvw] [-t vfstype] [-o options] device dir
Description	All files accessible in a Unix system are arranged in one big tree, the file hierarchy, rooted at /. These files can be spread out over several devices. The mount command serves to attach the file system found on some device to the big file tree.
Example	$mount /dev/fd0 /mnt/floppy

Command	umount
Syntax	umount [-dflnrv] dir | device [...]
Description	The umount command detaches the file system(s) mentioned from the file hierarchy. A file system is specified by giving the directory where it has been mounted.
Example	$umount /dev/fd0 /mnt/floppy

Command	find
Syntax	find [path...] [expression]
Description	This manual page documents the GNU version of find. find searches the directory tree rooted at each given file name by evaluating the given expression from left to right, according to the rules of precedence (see section OPERATORS), until the outcome is known (the left hand side is false for and operations, true for or), at which point find moves on to the next file name.
Example	$find . -name "*.java" ./OnlineStockTrading.java ./StockTrading.java

Command	umask
Syntax	umask value (octal)
Description	he umask is used by open(2) to set initial file permissions on a newly-created file. Specifically, permissions in the umask are turned off from the mode argument to open(2)
Example	$umask The common umask default value of 022 results in new files being created with permissions 0666 & ~022 = 0644 = rw-r--r-- in the usual case where the mode is specified as 0666

Command	ulimit
Syntax	ulimit <new Size of file…>
Description	It limits the resourses on file
Example	ulimit 1024

Command	ps
Syntax	ps [options]
Description	ps gives a snapshot of the current processes. If you want a repetitive update of this status, use top. This man page documents the /proc-based version of ps, or tries to.
Example	$ps –ef UID PID PPID C STIME TTY TIME CMD hope 29197 18961 0 Sep27 ? 00:00:06 sshd: hope@pts/87

Command	who
Syntax	who [OPTION]... [FILE | ARG1 ARG2]
Description	show who is logged on
Example	$who

Command	w
Syntax	w - [-husfV] [user]
Description	w displays information about the users currently on the machine, and their processes. The header shows, in this order, the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.
Example	$w

1.4 PRE LAB QUESTION

1. What is Open source software?

2. Write a command that will display all .txt files, including its individual permission.

1.5 LAB ASSIGNMENT

1. Write a command to display number of users logged in Linux operating system.

2. Write a command to create directory shprogs and /shprogs/weekprogs subdirectory.

1.6 POST LAB QUESTIONS

1. What are the kinds of permissions under Linux?

2. What are redirection operators?

EXPERIMENT - 2

BASIC COMMANDS-II

2.1 OBJECTIVE:

To Study and Practice on various commands like cat, tail, head , sort, nl, uniq, grep, egrep, fgrep, cut, paste, join, tee, pg, comm, cmp, diff, tr, awk, tar, cpio.

2.2 RESOURCES:

Linux operating system, vi-editor, shell-interpreter

2.3 DESCRIPTION / PROCEDURE

1. Open Linux Operating System Command Line Interface.
2. Execute command with options in shell prompt.
3. Press ctrl +z to exit from process.

Command	cat
Syntax	cat [OPTION] [FILE]...
Description	Concatenate FILE(s), or standard input, to standard output.
Example	$cat file4 Qwe Asdf Zxc

Command	tail
Syntax	tail [OPTION]... [FILE]...
Description	Print the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
Example	$tail file3 Zxc Zxcv Xcvb.... zxcv

Command	head
Syntax	head [OPTION]... [FILE]...
Description	Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
Example	$head file3 Asd Asd Awer.... Aqwe
Command	nl
Syntax	nl – [option] file…
Description	nl copies each specified file to the standard output, with line numbers added to the lines. The line number is reset to 1 at the top of each logical page. nl treats all of the

	input files as a single document and does not reset line numbers or logical pages between files.
Example	$nl file3 1 Asd 2 Asd .. .. 10 aqwe

Command	uniq
Syntax	uniq –[version] [-option] infile outfile
Description	uniq prints the unique lines in a sorted file, retaining only one of a run of matching lines. Optionally, it can show only lines that appear exactly once, or lines that appear more than once. uniq requires sorted input since it compares only consecutive lines.
Example	$uniq file7 Asd Zxc Qwe

Command	grep
Syntax	grep [options] PATTERN [FILE...]
Description	grep searches the named input FILEs (or standard input if no files are named, or the file name- is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.
Example	$grep UNIX example.txt UNIX operating system UNIX and Linux operating system

Command	egrep
Syntax	Egrep (grep –E) - grep [options] PATTERN [FILE...]
Description	-E, --extended-regexp Interpret PATTERN as an extended regular expression
Example	$ egrep 'Marketing|DBA' employee.txt 400 Nisha Manager Marketing $9,500 500 Randy DBA Technology $6,000

Command	fgrep
Syntax	grep –f [options] PATTERN [FILE...]
Description	Obtain patterns from FILE, one per line. The empty file contains zero patterns, and therefore matches nothing.
Example	$fgrep –f file3 file4

Command	cut
Syntax	cut [OPTION]... [FILE]...

Description	Print selected parts of lines from each FILE to standard output.
Example	$cut -c4 file.txt x u l

Command	paste
Syntax	paste [-delimiters=delim-list] [file...]
Description	paste prints lines consisting of sequentially corresponding lines of each specified file. In the output the original lines are separated by TABs. The output line is terminated with a newline.
Example	$paste –d “ “ file3 file4 Qwer tyui Asdd fgh Zxcv bnm n

Command	join
Syntax	join [OPTION]... FILE1 FILE2
Description	For each pair of input lines with identical join fields, write a line to standard output. The default join field is the first, delimited by whitespace. When FILE1 or FILE2 (not both) is -, read standard input.
Example	$join 1.txt 2.txt 1 abc abc 3 pqr lmn

Command	tee
Syntax	tee [OPTION]... [FILE]...
Description	Copy standard input to each FILE, and also to standard output.
Example	$ls|tee file7

Command	pg
Syntax	pg -[option] file list
Description	It makes the format to the given files
Example	$pg file7 file9 $pg myfile

Command	comm
Syntax	comm [OPTION]... LEFT_FILE RIGHT_FILE
Description	Compare sorted files LEFT_FILE and RIGHT_FILE line by line. -1 suppress lines unique to left file (Col 1) -2 suppress lines unique to right file(Col 2) -3 suppress lines that appear in both files(col 3)
Example	$ comm name_list.txt name_list_new.txt Bram Moolenaar Dennis Ritchie Ken Thompson Linus Torvalds Richard Stallman

Command	cmp
Syntax	cmp [-l | -s] file1 file2 [skip1 [skip2]]
Description	The cmp utility compares two files of any type and writes the results to the standard output. By default, cmp is silent if the files are the same; if they differ, the byte and line number at which the first difference occurred is reported. (In bytes)
Example	$cmp file10 file11 $_

Command	diff
Syntax	diff [options] from-file to-file
Description	In the simplest case, diff compares the contents of the two files from-file and to-file. A file name of - stands for text read from the standard input. As a special case, diff - - compares a copy of standard input to itself.
Example	$diff file8 file9 2a3,4 > Jean JRS@pollux.ucs.co > Jim jim@frolix8

Command	tr
Syntax	tr [OPTION]... SET1 [SET2]
Description	Translate, squeeze, and/or delete characters from standard input, writing to standard output.
Example	$tr –s “ ” < file3

Command	awk
Syntax	awk 'pattern {action}' input-file > output-file
Description	it allows the user to manipulate files that are structured as columns of data and strings.
Example	$awk '{ if($9 == "t4") print $0;}' input_file -rw-r--r-- 1 pcenter pcenter 43 Dec 8 21:39 t4

Command	tar
Syntax	tar –[options] [archive file] [files..]

Description	This manual page documents the GNU version of tar , an archiving program designed to store and extract files from an archive file known as a tarfile. A tarfile may be made on a tape drive, however, it is also common to write a tarfile to a normal file. The first argument to tar must be one of the options: Acdrtux, followed by any optional functions. The final arguments to tar are the names of the files or directories which should be archived. The use of a directory name always implies that the subdirectories below should be included in the archive.
Example	$tar -cvf archive.tar dir/

Command	cpio
Syntax	cpio - < name-list [> archive]
Description	This manual page documents the GNU version of cpio. cpio copies files into or out of a cpio or tar archive, which is a file that contains other files plus information about them, such as their file name, owner, timestamps, and access permissions. The archive can be another file on the disk, a magnetic tape, or a pipe.
Example	$ ls | cpio -ov > /tmp/object.cpio $cpio -idv < /tmp/object.cpio

2.4 PRE LAB QUESTION

1. What is grep command?

2. What is CLI?

2.5 LAB ASSIGNMENT

1. Write a sed command to check the length of a line from a text file.

2. Write a command to display files in given directory.

2.6 POST LAB QUESTIONS

1. How many shell scripts come with Linux operating system?

2. What are the three modes of operation of vi editor? Explain in brief.

EXPERIMENT - 3

SHELL PROGRAMMING - I

3.1 OBJECTIVE:

a) To write a shell script to print all .txt files and .c files
b) To write a shell script to move a set of files to a specified directory.
c) To write a shell script to display all the users who are currently logged in after a specified time.
d) To write a shell script to wish the user based on the login time.

3.2 RESOURCES:

Linux operating system, vi-editor, shell-interpreter

3.3 PROGRAM LOGIC:

Read a list of files from current directory and display output as for requirement.

3.4 DESCRIPTION / PROCEDURE

1. Open Linux Operating System Command Line Interface.
2. Open vi editor and type shell script.
3. Save file and exit from vi editor.
4. Execute shell script
5. Press ctrl +z to exit from process.

1. To write a shell script to print all .txt files and .c files

script: vi list.sh

```
echo list all text and c files to output stream

echo ls *.txt *.c
```

Input :

```
$sh list.sh
```

Output:

```
list all text and c files to output stream
fact.c, file.c, emp.txt
```

2. To write a shell script to move a set of files to a specified directory.

script: vi mov.sh

```
echo "enter source filename"
read sname
echo "enter directory name"
read dname
if [ -f $sname -a -d $dname ]
then
mv $sname $dname
else
echo "file or directory doesnt exists"
```

```
fi
```

Input :

```
$sh mov.sh
enter source filename
f1
enter directory name
orc
```

Output:

Output: file is moved to destination directory.

3. To write a Shell program to display all the users who are currently logged in after a specified time.

script: vi mov.sh

```
echo "enter time to list specified users who login after specified time"
read time1
for i in `who|tr -s " " "|"|cut -d "|" -f1`
do
t=`who|tr -s " " "|"|cut -d "|" -f4|cut -c1,2`
for s in $t
do
if [ $time1 -ge $s ]
then
echo $i
fi
done
done
```

Input :

```
$sh users.sh
enter time to list specified users who login after specified time
12
```

Output:

Iare10273

4. Write a Shell Program to wish the user based on the login time.

script: vi mov.sh

```
echo "displaying message based on login time"
hours=`who am i|tr -s " " "|"|cut -d "|" -f4|cut -c1,2`
  if [ $hours -le 12 ]
  then
      echo "Good Morning"
  else
      if [ $hours -le 16 ]
      then
          echo "Good Afternoon"
      elif [ $hours -le 20 ]
      then
          echo "Good Evening"
      else
          echo "Good Night"
      fi
   fi
```

Input :

$sh wish.sh

Output:

displaying message based on login time
Good Afternoon

3.5 PRE LAB QUESTION

1. Define shell script? What is the difference between shell and kernel.

2. Name few file handling commands present in unix.

3.6 LAB ASSIGNMENT

1. Write a shell script to count number of words present in a file without using commands.

2. Write a menu driven shell script to execute a command as 1.for `ls`, 2 for grep and 3 for cat.

3.7 POST LAB QUESTIONS

1. What is the purpose of case statement?
2. What the difference between break and exit statement?

EXPERIMENT - 4

SHELL PROGRAMMING - II

4.1 OBJECTIVE:

a) Write a Shell program to pass a message to a group of members, individual member and all.
b) Write a Shell program to count the number of words in a file.
c) Write a Shell program to calculate the factorial of a given number.
d) Write a Shell program to generate Fibonacci series.

4.2 RESOURCES:

Linux operating system, vi-editor, shell-interpreter

4.3 PROGRAM LOGIC:

Read a list of files from current directory and display output as for requirement.

4.4 DESCRIPTION / PROCEDURE

1. Open Linux Operating System Command Line Interface.
2. Open vi editor and type shell script.
3. Save file and exit from vi editor.
4. Execute shell script
5. Press ctrl +z to exit from process.

1. To write a Shell program to pass a message to a group of members, individual member and all.

script: vi message.sh

```
echo "Enter the choice to send the message 1- Group,2-Individual,3-All,4.invalid"

read  choice

echo "enter the message"

read msg

case $choice in

        1)write $* $msg ;;

        2)echo "enter the username"

          read  username

          write $username  $msg;;

        3)wall  $msg ;;

        *)echo "Invalid Entry"

esac
```

Input :

$sh list.sh

Output: based on choice.

2. To Write a Shell program to count the number of words in a file.

script: vi wc1.sh

```
echo "displaying number of words of given file"
echo "enter source filename"
read sname
if [ -f $sname ]
then
wc -l $sname
else
echo "file doesnt exists"
fi
```

Input :

$sh wc1.sh
enter source filename
f1

Output:

displaying number of words of given file.
23 f1

3. To Write a Shell program to calculate the factorial of a given number.

script: vi fact.sh

```
i=2
res=1
echo "enter number to find factorial"
read num
if [ $num -ge 2 ]
then
while [ $i -le $num ]
do
res=`expr $res \* $i`
i=`expr $i + 1`
done
fi
echo factorial of given number is $res
```

Input :

$sh fact1.sh
enter number to find factorial
5

Output:

factorial of given number is 120

4. To write a Shell program to generate Fibonacci series.

script: vi fib.sh

```
echo -n "Enter How many numbers:"
```

```
read num
num1=0
num2=1
echo -n "Fibonacci series: "
echo -n "$num1"
echo -n " $num2 "
count=2
while [ $count -lt $num ]
do
num3=`expr $num1 + $num2`
echo -n " $num3 "
num1=$num2
num2=$num3
count=`expr $count + 1`
done
```

Input :

$sh fib.sh

Output:

Enter How many numbers:
5
Fibonacci series 0 1 1 2 3

4.5 PRE LAB QUESTION

1. What are positional parameter and name any two.
2. Write down the syntax of `if` statement.

4.6 LAB ASSIGNMENT

1. Read two string str1 and str2 and check

   i) Compare two strings
   ii) Palindrome or not

4.7 POST LAB QUESTIONS

1. What is the purpose of the variable $? What are the various output it has?

EXPERIMENT - 5

SIMULATING COMMANDS I

5.1 OBJECTIVE:

a) Simulate cat command
b) Simulate cp command

5.2 RESOURCES:

Linux operating system, vi-editor, C compiler

5.3 PROGRAM LOGIC:

Read a list of arguments and implement commands using system calls.

5.4 DESCRIPTION / PROCEDURE

1. Open Linux Operating System Command Line Interface.
2. Open vi editor and type program.
3. Save file and exit from vi editor.
4. Execute c program.
5. Press ctrl +z to exit from process.

1. To write a program to simulate cat command.

script: vi cat1.c

```
#include<fcntl.h>
#include<sys/stat.h> #define BUFSIZE 1
int main(int argc, char **argv)
{
int fd1; int n; char buf;
fd1=open(argv[1],O_RDONLY);  printf("Displaying content of file\n");
while((n=read(fd1,&buf,1))>0)
{
printf("%c",buf); /* or
write(1,&buf,1);  */
}
return (0);
}
```

Input :

$ cc prog11a.c unit1

Output:

Displays content of file

2. To Write a program to simulate cp command

script: vi cp1.c

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<string.h>
```

```
void main()
{
char src[10], dest[10], buff; int fd,fd1;
printf("enter the source file name \n"); scanf("%s\n",src); fd=open("src",O_RDONLY); printf("enter the destination
file name\n" scanf("%s\n",dest);
fd1=open("dest",O_WRONLY|O_CREAT|O_TRUNC|S_IRUSR|S_IWUSR); while(read(fd,&buff,1));
wirte(fd1,&buff,1);
printf("The copy of a file is successed"); close(fd);
close(fd1); }
```

Input :

cc prog10.c
./a.out
entr the source file name: file1
enter the destination file name: file2

Output:

The copy of a file is successes

5.5 PRE LAB QUESTION

1. What is the difference between $* and $@.
2. How to read a variable ,assign ,and access it

5.6 LAB ASSIGNMENT

1. Read a file name from command line and check it‟s a file or not.
2. Read a file name from command line and check if it read and write permission or not.

5.7 POST LAB QUESTIONS

1. Explain how to check file is existing or not, it has read, write and execution permissions or not.

EXPERIMENT - 6

SIMULATING COMMANDS II

6.1 OBJECTIVE:

a) Simulate tail command
b) Simulate head command

6.2 RESOURCES:

Linux operating system, vi-editor, C compiler

6.3 PROGRAM LOGIC:

Read a list of arguments and implement commands using system calls.

6.4 DESCRIPTION / PROCEDURE

6. Open Linux Operating System Command Line Interface.
7. Open vi editor and type program.
8. Save file and exit from vi editor.
9. Execute c program.
10. Press ctrl +z to exit from process.

1. To write a program to simulate tail command.

script: vi tail1.c

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
 FILE *fp;
 char str[80];
 int n,i=0;
 printf("enter the number of lines need to print");
 scanf("%d",&n);
 fp=fopen("head.c","r");
 while(!feof(fp))
    {fgets(str,80,fp);
      i++;
    }
rewind(fp);
n=i-n;
i=0;
while(!feof(fp))
   {      fgets(str,80,fp);
      i++;
```

```
        if(i>n)
        printf("%s",str);
      }
    getch();}
```

Input :

cc tail1.c
Displaying content of file
5

Output:

Displays content of file last 5 lines

2. To Write a program to simulate head command

script: vi head1.c

```
#include<stdio.h>
#include<string.h>
main()
{
 FILE *fp;
 char str[80];
 int n,i=0;
 printf("enter the number of lines need to print");
 scanf("%d",&n);
 fp=fopen("fact1.sh","r");
 while(!feof(fp))
 {
      fgets(str,80,fp);
      i++;
      printf("%d %s",str);

  if(i==n)
  break;
 }
 }
```

Input :

cc head1.c
./a.out
enter the number of lines need to print
5

Output:

Displays first 5 lines to output stream

6.5 PRE LAB QUESTION

1. What is meant by file descriptor and user file descriptor starts from which number

6.6 LAB ASSIGNMENT

1. Write a c-program to count number lines in a file.

6.7 POST LAB QUESTIONS

1. What are the file descriptors values of keyword, monitor, error.
2. What is the use of lseek() function

EXPERIMENT - 7

SIMULATING COMMANDS III

7.1 OBJECTIVE:

a) Simulate mv command
b) Simulate nl command

7.2 RESOURCES:

Linux operating system, vi-editor, C compiler

7.3 PROGRAM LOGIC:

Read a list of arguments and implement commands using system calls.

7.4 DESCRIPTION / PROCEDURE

11. Open Linux Operating System Command Line Interface.
12. Open vi editor and type program.
13. Save file and exit from vi editor.
14. Execute c program.
15. Press ctrl +z to exit from process.

1. To write a program to simulate mv command.

script: vi mv1.c

```
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
main()
{
int  fd,n;
char c;
fd1=open("abc.c",O_RDONLY,0644);
fd2=open("xyz.c",O_WRONLY,0644);
while((n=read(fd1,&c,1))>0)
write(fd2,&c,n);
unlink("abc.c")
}
```

Input :

cc mv1.c

Output:

Success if source file exists

2. To Write a program to simulate nl command

script: vi nl1.c

```
#include<stdio.h>
```

```
#include<string.h>
main()
{
 FILE *fp;
 char str[80];
 int i=0;
fp=fopen("fact1.sh","r");
 while(!feof(fp))
 {
      fgets(str,80,fp);
      i++;
      printf("%d %s",i,str);
  }
 }
```

Input :

cc nl1.c

./a.out

enter the number of lines need to print

5

Output:

Displays first 5 lines to output stream with line number

7.5 PRE LAB QUESTION

1. What is the difference between open() and fopen()?

7.6 LAB ASSIGNMENT

1. Write a c-program to count number words in a file.

7.7 POST LAB QUESTIONS

1. What is the difference between read(), write() and scanf, printf respectively

EXPERIMENT - 8

SIGNAL HANDLING

8.1 OBJECTIVE:

Write a program to handle the signals like SIGINT, SIGDFL, SIGIGN

8.2 RESOURCES:

Linux operating system, vi-editor, C compiler

8.3 PROGRAM LOGIC:

Read a list of arguments and handle signals using system calls.

8.4 DESCRIPTION / PROCEDURE

16. Open Linux Operating System Command Line Interface.
17. Open vi editor and type program.
18. Save file and exit from vi editor.
19. Execute c program.
20. Press ctrl +z to exit from process.

1. To write a program to simulate mv command.

Program:

a) #include <signal.h>

```
#include <stdio.h>
#include <unistd.h>
        int x = 1;
        void intr(int sig) {
        printf("dividing by zero!\n");
        x = 0;
        }
void fpe(int sig) {
        printf("FPE!  I got a signal: %d\n",sig);
        psignal(sig, "psignal");
        x = 1;
}

 int main(void) {
        (void) signal(SIGINT, intr);
        (void) signal(SIGFPE, fpe);
        while(1)
        {
        printf("Hello World: %d\n",1/x);
        sleep(1);
        }
}
```

b)

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

void intr(int sig) {
        printf("Interupted");
        exit(1);
}
int main(void) {
        (void) signal(SIGINT, intr);

        while(1)
        {
        printf("to stop press /^c Cntrl + c”);
        sleep(1);
         }
}
```

c)

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>

void q(int sig) {
printf("Interupted");
        exit(1);
                        }
int main(void) {
(void) signal(SIGQUIT, q);

        while(1)
        {
        printf("to stop press Ctl^\ Cntrl + \”);
        sleep(1);
        }
}
```

8.5 PRE LAB QUESTION

1. What is the difference between open() and fopen()?

8.6 LAB ASSIGNMENT

1. Write a c-program to count number words in a file.

8.7 POST LAB QUESTIONS

1. What is the difference between read(), write() and scanf, printf respectively

EXPERIMENT - 9

INTERPROCESS COMMUNICATIONS I

9.1 OBJECTIVE:

To Write a C program to implement the following IPC forms
a) FIFO b) PIPE

9.2 RESOURCES:

Linux operating system, vi-editor, C compiler

9.3 PROGRAM LOGIC:

Read a list of arguments and exchange data between processes using system calls.

9.4 DESCRIPTION / PROCEDURE

21. Open Linux Operating System Command Line Interface.
22. Open vi editor and type program.
23. Save file and exit from vi editor.
24. Execute c program.
25. Press ctrl +z to exit from process.

Program:

a) To write a program to implement PIPE IPC.

--sending----

```
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
   int fd;
   char * myfifo = "/tmp/myfifo";

   /* create the FIFO (named pipe) */
   mkfifo(myfifo, 0666);

   /* write "Hi" to the FIFO */
   fd = open(myfifo, O_WRONLY);
   write(fd, "Hi", sizeof("Hi"));
   close(fd);
   unlink(myfifo);
   return 0;
}
```

---reciving----

```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <unistd.h>

#define MAX_BUF 1024

int main()
{
   int fd;
   char * myfifo = "/tmp/myfifo";
   char buf[MAX_BUF];

   /* open, read, and display the message from the FIFO */
   fd = open(myfifo, O_RDONLY);
   read(fd, buf, MAX_BUF);
   printf("Received: %s\n", buf);
   close(fd);

   return 0;
}
```

b) To write a program to implement PIPE IPC.

```
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

main()
{
int pid,pfd[2],n,a,b,c;
if(pipe(pfd)==-1)
{
printf("\nError in pipe connection\n");
exit(1);
}
pid=fork();
if(pid>0)
{
printf("\nParent Process");\
printf("\n\n\tFibonacci Series");
printf("\nEnter the limit for the series:");
scanf("%d",&n);
close(pfd[0]);
write(pfd[1],&n,sizeof(n));
close(pfd[1]);
exit(0);
}
else
{
close(pfd[1]);
read(pfd[0],&n,sizeof(n));
printf("\nChild Process");
```

```
a=0;
b=1;
close(pfd[0]);
printf("\nFibonacci Series is:");
printf("\n\n%d\n%d",a,b);
while(n>2)
{
c=a+b;
printf("\n%d",c);
a=b;
b=c;
n--;
}}}
```

9.5 PRE LAB QUESTION

1. What are process identifiers in Linux programming.
2. What is process, how you create new process?

9.6 LAB ASSIGNMENT

1. Write a program to find sum of odd numbers of parent process and sum of even numbers by child process.

9.7 POST LAB QUESTIONS

1. Illustrate difference between fork() and vfork() functions.
2. What are different process ids in Linux programming?

EXPERIMENT - 10

MESSAGE QUEUES

10.1 OBJECTIVE:

1. To write a C program (sender.c) to create a message queue with read and write permissions to write 3 messages to it with different priority numbers.
2. To write a C program (receiver.c) that receives the messages (from the above message queue as specified and displays them.

10.2 RESOURCES:

Linux operating system, vi-editor, C compiler

10.3 PROGRAM LOGIC:

Read a list of arguments and exchange data between processes using system calls.

10.4 DESCRIPTION / PROCEDURE

26. Open Linux Operating System Command Line Interface.
27. Open vi editor and type program.
28. Save file and exit from vi editor.
29. Execute c program.
30. Press ctrl +z to exit from process.

Program:

a) To write a C program (sender.c) to create a message queue with read and write permissions to write 3 messages to it with different priority numbers.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <string.h>
#define MSGSZ     128
typedef struct msgbuf {
        long    mtype;
        char    mtext[MSGSZ];
        } message_buf;

        main()
        {
        int msqid;
        int msgflg = IPC_CREAT | 0666;
        key_t key;
         message_buf sbuf;
        size_t buf_length;
         key = 1234;
```

```
(void) fprintf(stderr, "\nmsgget: Calling msgget(%#lx,\%#o)\n",key, msgflg);

if ((msqid = msgget(key, msgflg )) < 0) {
perror("msgget");
exit(1);
}
else
(void) fprintf(stderr,"msgget: msgget succeeded: msqid = %d\n", msqid);
 sbuf.mtype = 1;
   (void) fprintf(stderr,"msgget: msgget succeeded: msqid = %d\n", msqid);
(void) strcpy(sbuf.mtext, "Did you get this?");
(void) fprintf(stderr,"msgget: msgget succeeded: msqid = %d\n", msqid);
buf_length = strlen(sbuf.mtext) + 1 ;
if (msgsnd(msqid, &sbuf, buf_length, IPC_NOWAIT) < 0) {
printf ("%d, %d, %s, %d\n", msqid, sbuf.mtype, sbuf.mtext, buf_length);
perror("msgsnd");
exit(1);
 }

else
printf("Message: \"%s\" Sent\n", sbuf.mtext);
exit(0);
}
```

b) To write a C program (receiver.c) that receives the messages (from the above message queue as specified and displays them.

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#define MSGSZ     128
typedef struct msgbuf {
   long    mtype;
   char    mtext[MSGSZ];
   } message_buf;
          main()
          {
   int msqid;
   key_t key;
   key = 1234;
   if ((msqid = msgget(key, 0666)) < 0) {
      perror("msgget");
      exit(1);
   }
   if (msgrcv(msqid, &rbuf, MSGSZ, 1, 0) < 0) {
      perror("msgrcv");
      exit(1);
   }
   printf("%s\n", rbuf.mtext);
   exit(0);}
```

10.5 PRE LAB QUESTION

1. What is the purpose of msgget(),msgsnd(),msgrcv().
2. What is structure of message queue.

10.6 LAB ASSIGNMENT

1. Implement message queues like sender and receiver, where receiver can receive the message in un-order.

10.7 POST LAB QUESTIONS

1. Describe use of pipe, fifos and message queues.

EXPERIMENT - 11

SHARED MEMORY

11.1 OBJECTIVE:

1. To write a C program to implement shared memory form of IPC.

11.2 RESOURCES:

Linux operating system, vi-editor, C compiler

11.3 PROGRAM LOGIC:

Read a list of arguments and exchange data between processes using system calls.

11.4 DESCRIPTION / PROCEDURE

31. Open Linux Operating System Command Line Interface.
32. Open vi editor and type program.
33. Save file and exit from vi editor.
34. Execute c program.
35. Press ctrl +z to exit from process.

Program:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ     27
main()
{    char c;
    int shmid;
    key_t key;
    char *shm, *s;
    key = 5678;
    if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0)
 {        perror("shmget");
        exit(1);     }
    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1)
 {        perror("shmat");
        exit(1);     }
    s = shm;
    for (c = 'a'; c <= 'z'; c++)
        *s++ = c;
    *s = NULL;
    while (*shm != '*')
        sleep(1);
    exit(0);
}
```

shm_client.c

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define SHMSZ     27
main()
{
   int shmid;
   key_t key;
   char *shm, *s;
    key = 5678;
   if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {
      perror("shmget");
      exit(1);    }
   if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
      perror("shmat");
      exit(1);    }
   for (s = shm; *s != NULL; s++)
      putchar(*s);
   putchar('\n');

   *shm = '*';

   exit(0);
}
```

11.5 LAB VIVA QUESTIONS:

1. How many message queues can create in linux programming and what are limitations for message queues?

2. Explain how to handle shared memory in IPC.

11.6 PRE LAB QUESTION

1. Implement message queues like sender and receiver, where sender sends number and receiver can receive the message and find square of received number.

11.7 POST LAB QUESTIONS

1. Discuss how to attach and detach to shared memory.

EXPERIMENT - 12

SOCKET PROGRAMMING

12.1 OBJECTIVE:

1. To write client and server programs (using c) for interaction between server and client processes using TCP Elementary functions.
2. To write client and server programs (using c) for interaction between server and client processes using UDP Elementary functions.

12.2 RESOURCES:

Linux operating system, vi-editor, C compiler

12.3 PROGRAM LOGIC:

Read a list of arguments and exchange data between processes using system calls.

12.4 DESCRIPTION / PROCEDURE

1. Open Linux Operating System Command Line Interface.
2. Open vi editor and type program.
3. Save file and exit from vi editor.
4. Execute c program.
5. Press ctrl +z to exit from process.

Program:

1. To write client and server programs (using c) for interaction between server and client processes using TCP Elementary functions.

client.c

```
#include<stdio.h>
#include<unistd.h>
#include<sys/socket.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
main()
{
	int lfd,cfd;
	char fub[10];
	struct sockaddr_in sa;
	lfd=socket(AF_INET,SOCK_STREAM,0);
	printf("Socket was created\n");
	sa.sin_family=AF_INET;
	sa.sin_port=htons(61239);
	sa.sin_addr.s_addr=htonl(0L);
	bind(lfd,(struct sockaddr*)&sa,sizeof(sa));
	printf("Bind completed\n");
	listen(lfd,3);
	cfd=accept(lfd,0,0);
	printf("Accepted\n");
```

```
        read(cfd,fub,10);
        printf("Read completed\n");
        write(cfd,fub,10);
        printf("Write completed\n");
}
```

Server.c

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<sys/socket.h>
main()
{
        int sfd,d;
        char buf[10];
        struct sockaddr_in sa;
        sfd=socket(AF_INET,SOCK_STREAM,0);
        printf("Socket was created\n");
        sa.sin_family=AF_INET;
        sa.sin_port=htons(61239);
        sa.sin_addr.s_addr=htons(0L);
        d=connect(sfd,(struct sockaddr*)&sa,sizeof(sa));
        if(d==0)
        {
                printf("Connected\n");
                write(sfd,"Hello",6);
                printf("Write returns : ");
                read(sfd,buf,6);
                printf("%s\n",buf);
        }
        else
        {
                printf("Not yet connected\n");
                sleep(10);
        }
}
```

2. To write client and server programs (using c) for interaction between server and client processes using UDP Elementary functions.

Client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <fcntl.h>
```

```
#include <signal.h>
#include <unistd.h>
#define BUFSIZE 512
static void sig_usr(int);
void str_cli(FILE *fp , int sockfd , struct sockaddr *server , socklen_t len);
int main( int C, char *argv[] )
{
int sd;
Struct sockaddr_in serveraddress;

/*installing signal Handlers*/

signal (SIGPIPE,sig_usr);
signal (SIGINT,sig_usr);
if (NULL==argv[1])
{
printf("Please enter the IP Address of the server\n");
exit(0);
}
if (NULL==argv[2])
{
printf("Please enter the Port Number of the server\n");
exit(0);
}
sd = socket( AF_INET, SOCK_DGRAM, 0 );
if( sd < 0 )
{
perror( "socket" );
exit( 1 );
}
memset( &serveraddress, 0, sizeof(serveraddress) );
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(atoi(argv[2]));//PORT NO
serveraddress.sin_addr.s_addr = inet_addr(argv[1]);//ADDRESS
printf("Client Starting service\n");
printf("Enter Data For the server\n");
str_cli(stdin,sd ,(struct sockaddr *)&serveraddress,
sizeof(serveraddress));
}
```

Server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <signal.h>
#define BUFSIZE 512
#define MYPORT 11710
#define MAXNAME 100
int main(int C, char **V )
{
int sd,n,ret;
struct sockaddr_in
38
serveraddress,cliaddr;
socklen_t length;
char clientname[MAXNAME],datareceived[BUFSIZE];
sd = socket( AF_INET, SOCK_DGRAM, 0 );
if( sd < 0 ) {
perror( "socket" );
exit( 1 );
}
memset( &serveraddress, 0, sizeof(serveraddress) );
memset( &cliaddr, 0, sizeof(cliaddr) );
serveraddress.sin_family = AF_INET;
serveraddress.sin_port = htons(MYPORT);//PORT NO
serveraddress.sin_addr.s_addr = htonl(INADDR_ANY);//IP ADDRESS
ret=bind(sd,(struct sockaddr*)&serveraddress,sizeof(serveraddress));
if(ret<0)
{
perror("BIND FAILS");
exit(1);
}
for(;;)
{
printf("I am waiting\n");
/*Received a datagram*/
length=sizeof(cliaddr);
n=recvfrom(sd,datareceived,BUFSIZE,0,
(struct sockaddr*)&cliaddr , &length);
printf("Data Received from %s\n",
inet_ntop(AF_INET,&cliaddr.sin_addr,
clientname,sizeof(clientname)));
/*Sending the Received datagram back*/
datareceived[n]='\0';
printf("I have received %s\n",datareceived);
sendto(sd,datareceived,n,0,(struct sockaddr *)&cliaddr,length);
}
```

}

12.5 LAB VIVA QUESTIONS:

1. List wellknown ports for TCP and UDP.

2. What is the purpose of connect and bind function in socket?

12.6 PRE LAB QUESTION

1. Write a program to design a TCP client – server application which takes IP address, Port number and string to be echoed as command line inputs in client application and implements echo service.

12.7 POST LAB QUESTIONS

1. Explain about IPV6 socket address structure and compare it with IPV4 and unix socket address structures.