LINUX PROGRAMMING

LAB MANUAL

Subject Code:	57609
Regulations:	R13 – JNTUH
Class:	IV Year I Semester (CSE)

Prepared By

K. Radhika
Associate Professor

Department of Computer Science & Engineering

INSTITUTE OF AERONAUTICAL ENGINEERING

Dundigal – 500 043, Hyderabad

INSTITUTE OF AERONAUTICAL ENGINEERING

Dundigal, Hyderabad - 500 043

COMPUTER SCIENCE AND ENGINEERING

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	**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.
PO12	**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.

Program Specific Outcomes

PSO1	**Professional Skills:** The ability to research, understand and implement 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	**Problem-Solving Skills:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product 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.

LINUX PROGRAMMING LAB SYLLABUS

ATTAINMENT OF PROGRAM OUTCOMES & PROGRAM SPECIFIC OUTCOMES

Exp. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
	LINUX PROGRAMMING		
1	c) Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers. d) *Illustrate by writing script that will print, message "Hello World, in Bold and Blink effect, and in different colors like red, brown etc using echo commands?	PO1, PO2	PSO1
2	c) Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it. d) *Illustrate by writing script using for loop to print the following patterns? i) * ii) 1 * * 2 2 * * * 3 3 3 * * * * 4 4 4 4 * * * * * 5 5 5 5 5	PO1	PSO1
3	c) Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions. d) * Illustrate to redirect the standard input (stdin) and the standard output (stdout) of a process, so that scanf () reads from the pipe and printf () writes into the pipe?	PO1, PO2	PSO1
4	c) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported. d) *Illustrate by writing c program where process forks to a child, and create a child process by using forks and suddenly terminates itself?	PO1	PSO1
5	Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.	PO1	PSO1
6	Write a shell script to list all of the directory files in a directory.	PO1	PSO1
7	Write a shell script to find factorial of a given integer.	PO1	PSO1
8	Write an awk script to count the number of lines in a file that do not contain vowels.	PO1	PSO1
9	Write an awk script to find the number of characters, words and lines in a file.	PO1	PSO1
10	Write a c program that makes a copy of a file using standard I/O and system calls	PO1	PSO1

Exp. No.	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
11	Implement in C the following UNIX commands using System calls a) cat b) ls c) mv	PO1, PO2	PSO1
12	Write a program that takes one or more file/directory names as command line input and reports the following information on the file. a) File type b) Number of links c) Time of last access d) Read Write and Execute permissions	PO1, PO2	PSO1
13	Write a C program to emulate the UNIX ls –l command.	PO1	PSO1
14	Write a C program to list for every file in a directory, its inode number and file name.	PO1	PSO1
15	Write a C program that demonstrates redirection of standard output to a file.Ex: ls > f1.	PO1	PSO1
16	Write a C program to create a child process and allow the parent to display “parent” and the child to display “child” on the screen.	PO1	PSO1
17	Write a C program to create a Zombie process.	PO1	PSO1
18	Write a C program that illustrates how an orphan is created.	PO1	PSO1
19	Write a C program that illustrates how to execute two commands concurrently with a command pipe. Ex: - ls –l | sort	PO1	PSO1
20	Write C programs that illustrate communication between two unrelated processes using named pipe.	PO1	PSO1
21	Write a C program to create a message queue with read and write permissions to write 3 messages to it with different priority numbers.	PO1	PSO1
22	Write a C program that receives the messages (from the above message queue as specified in (21)) and displays them.	PO1	PSO1
23	Write a C program to allow cooperating processes to lock a resource for exclusive use, using a) Semaphores b) flock or lockf system calls.	PO1, PO2	PSO1
24	Write a C program that illustrates suspending and resuming processes using signals	PO1, PO2	PSO1
25	Write a C program that implements a producer-consumer system with two processes.	PO1, PO2, PO4	PSO1, PSO2
26	Write client and server programs (using c) for interaction between server and client processes using Unix Domain sockets. (Using Semaphores).	PO1, PO2, PO3, PO4	PSO1, PSO2
27	Write client and server programs (using c) for interaction between server and client processes using Internet Domain sockets.	PO1, PO2, PO3, PO4	PSO1, PSO2
28	Write a C program that illustrates two processes communicating using shared memory.	PO1, PO2, PO3, PO4	PSO1, PSO2

LINUX PROGRAMMING LABORATORY

OBJECTIVE:

The Linux programming 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.

Data mining tools allow predicting future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The data mining laboratory course is designed to exercise the data mining techniques such as classification, clustering, pattern mining etc with varied datasets and dynamic parameters. Weka data mining tool is used for the purpose of acquainting the students with the basic environment of the data mining tools.

OUTCOMES:

Upon the completion of Linux Programming and Data Mining practical course, the student will be able to:

1. **Understand** and implement basic system functionalities of Linux operating system.
2. **Write** shell scripts to automate different tasks.
3. **Demonstrate** Inter process Communication techniques.
4. **Design** and implement advanced features such as threads, IPC, pipes, FIFOs.
5. **Demonstrate** client server technology using socket programming.
6. **Understand** synchronized programs using shared memory,
7. **Implement** and manage client server technology using TCP and UDP.
8. **Learn** to build a data warehouse and query it using open source tools.
9. **Learn** to execute data mining tasks using a data mining toolkit (such as WEKA) and visualize the results.
10. **Demonstrate** the working of algorithms for data mining tasks such association rule mining, classification, clustering and regression.

LINUX PROGRAMMING

EXPERIMENT 1

1.1 OBJECTIVE

a) Write a shell script that accepts a file name, starting and ending line numbers as arguments and displays all the lines between the given line numbers.

b) *Illustrate by writing script that will print, message “Hello World, in Bold and Blink effect, and in different colors like red, brown etc using echo commands?

1.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

1.3 PROGRAM LOGIC

1. Read a filename, starting and ending line numbers as arguments
2. Find difference between starting and ending line numbers
3. Test given filename exists or not
4. If exists display between lines to output stream else display file not exists

1.4.a DESCRIPTION / PROCEDURE

```
echo " Enter the file name”
read fname
echo “enter starting line number”
read sl
echo “enter ending line number”
read el
d=`expr $el - $sl`
if [ -f $fname ]
   then
   echo "the lines between $sl and $el of given file are"
   head -$el $fname | tail -$d
   else
   echo "file doesnt exist"
fi
```

INPUT: sh
prog1.sh

```
enter the file
name file1
enter starting line number
15
enter ending line number
20
```

OUTPUT:

# It displays 15 to 20 between lines

1.4.b DESCRIPTION / PROCEDURE

```
# clear the screen
clear
echo -e "\033[1m Hello World"
# print bold effect
echo -e "\033[5m Blink"
 # blink effect
echo -e "\033[0m Hello World"
# print back to normal
```

```
echo -e "\033[31m Hello World"
#print in Red color
echo -e "\033[32m Hello World"
# Green color
echo -e "\033[33m Hello World"
# See remains on screen
echo -e "\033[34m Hello World"
echo -e "\033[35m Hello World"
echo -e "\033[36m Hello World"
echo -e -n "\033[0m "
 # print back to normal
echo -e "\033[41m Hello World"
echo -e "\033[42m Hello World"
echo -e "\033[43m Hello World"
echo -e "\033[44m Hello World"
echo -e "\033[45m Hello World"
echo -e "\033[46m Hello World"
echo -e "\033[0m Hello World"
# Print back to normal
```

1.5 PRE-LAB QUESTIONS

1. Define shell script? What is the difference between shell and kernel. 2. Name few file handling commands present in unix.

1.6 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.

1.7 POST-LAB QUESTIONS

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

EXPERIMENT 2

2.1 OBJECTIVE

a) Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it.

b) *Illustrate by writing script using for loop to print the following patterns?

2.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

2.3 PROGRAM LOGIC

Read file name from command line arguments and display lines inverse of specified word.

2.4.a DESCRIPTION / PROCEDURE

```
if [ $# -ne 0 ]
then
      echo enter the word
      read word
      for fname in
          $* do
          if [ -f $fname ]
             then
             echo the given input filename is:$fname
             grep -v "$word" $fname
          else
            echo its not a
          file fi
         done
          else
          echo "enter atleast one argument as input"
fi
```

INPUT:

```
sh prog2.sh 3.sh
enter the word
echo
```

OUTPUT:

```
The given input filename is : 3.sh
It displays all the lines other than pattern matching
```

2.4.b.i DESCRIPTION / PROCEDURE

```
# do    the    following    for
 loop echo "Stars"
# outer loop
 for (( i=1; i<=5; i++ ))
do
#inner loop

    for (( j=1; j<=i;  j++ ))
    do
     echo -n " *"
    done
    echo ""
done
 #
```

2.4.b.ii DESCRIPTION / PROCEDURE

```
# do the following for loop

echo "Can you see the following:"
# outer loop
for (( i=1; i<=5; i++ ))
do
#inner loop
    for (( j=1; j<=i; j++ ))
    do
     echo -n "$i"
    done
    echo ""
done
#
```

2.5 PRE-LAB QUESTIONS

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

2.6 LAB ASSIGNMENT

1.Read two string str1 and str2 and check
  i) Compare two strings
  ii) Palindrome or not

2.7 POST-LAB QUESTIONS

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

EXPERIMENT 3

3.1 OBJECTIVE

a) Write a shell script that displays a list of all the files in the current directory to which the user has read, write and execute permissions.

b) Illustrate to redirect the standard input (stdin) and the standard output (stdout) of a process, so that scanf () reads from the pipe and printf () writes into the pipe?

3.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

3.3 PROGRAM LOGIC

Read a list of files from current directory and display the file names to output stream whose files has read, write, execute permissions.

3.4.a DESCRIPTION / PROCEDURE

```
echo "List of Files which have Read, Write and Execute Permissions in Current
Directory" for file in *
 do
 if [ -r $file -a -w $file -a -x $file
    ] then
    echo $file
 fi
done
```

INPUT: sh
prog3.sh

OUTPUT:

```
List of Files which have Read, Write and Execute Permissions in Current
Directory pp2.txt
```

3.4.b DESCRIPTION / PROCEDURE

```
#include <stdio.h>
#include <unistd.h>
#include <sys/ipc.h>
main()
{
int fd[2];int n=0, i;
pipe(fd);
if (fork() == 0) {
/* create Child process */
close(1) ; dup(fd[1]) ;
close(fd[0]);
/* try not read from the pipe in this example.So close fd[0]. */ for
(i=0; i < 10; i++) {printf("%d\n",n);
/* Now that stdout has been      redirected, printf automatically writes into the pipe. */  n++;  }
} else {/* Parent process */close(0) ; dup(fd[0]) ;
/* Redirect the stdin of this   process to the pipe*/
close(fd[1]);
/* will not write into the pipe.So we close fd[1]. */
for (i=0; i < 10; i++) {scanf("%d",&n);
/* Now that stdin has been      redirected, scanf automatically reads from the pipe. */
printf("n = %d\n",n);
/* try stdout of this has not changed . So this will be shown in the terminal.      */  sleep(1);
}}}
```

3.5 PRE-LAB QUESTIONS

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

3.6 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.

3.7 POST-LAB QUESTIONS

1. How to check if file is existing, it has read, write and execution permission.

EXPERIMENT 4

4.1 OBJECTIVE

a) Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. Whenever the argument is a file, the number of lines on it is also reported.

b) *Illustrate by writing c program where process forks to a child, and create a child process by using forks and suddenly terminates itself?

4.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

4.3 PROGRAM LOGIC

Read a list of files from current directory and display the file names to output along with number of lines of each file

4.4.a DESCRIPTION / PROCEDURE

```
echo enter the name
for fname in
  * do
 if test -f
   $fname then
   echo "file" $fname
   echo "number of lines" `cat $fname | wc -
l` else if test -d $fname
   then
   echo "dir" $fname
 fi
fi
done
```

INPUT: sh
prog4.sh

OUTPUT:

```
enter the name
file 3.sh
number of lines 9
```

4.4.b DESCRIPTION / PROCEDURE

```
#include <stdio.h>
 /* for fork() */
#include <sys/types.h>
#include <unistd.h>
 /* for wait*() */
#include <sys/wait.h>
int main() {
   pid_t mypid,
   childpid; int status;
   mypid = getpid();
   printf("Hi. I'm the parent process. My pid is %d.\n", mypid);
   childpid = fork();
```

```
    if ( childpid == -1 ) {
        perror("Cannot proceed. fork() error");
        return 1;
    }
    if (childpid == 0) {
       printf("Child 1: I inherited my parent's pid as %d.\n", mypid);
        mypid = getpid();
        printf("Child 1: getppid() tells my parent is %d. My own pid instead is %d.\n", getppid(), mypid);
        /* forks another child */
        childpid = fork();
        if ( childpid == -1 ) {
            perror("Cannot proceed. fork() error");
            return 1;
        }
        if (childpid == 0) {
       /* this is the child of the first child, thus "Child 2" */
            printf("Child 2: I hinerited my parent's PID as %d.\n", mypid);
            mypid = getpid();
            printf("Child 2: getppid() tells my parent is %d. My own pid instead is %d.\n", getppid(),
                                                                 mypid);
            childpid = fork();
            if ( childpid == -1 ) {
                perror("Cannot proceed. fork() error");
                return 1;
            }
            if (childpid == 0) {
             /* "Child 3" sleeps 30 seconds then terminates 12, hopefully before its parent "Child 2" */
                printf("Child 3: I hinerited my parent's PID as %d.\n", mypid);
                mypid = getpid();
                printf("Child 3: getppid() tells my parent is %d. My own pid instead is %d.\n", getppid(),
                            mypid);
                sleep(30);
                return 12;
            } else /* the parent "Child 2" suddendly returns 15 */ return 15; }
        else {
             /* this is still "Child 1", which waits for its child to exit */
            while ( waitpid(childpid, &status, WNOHANG) == 0 ) sleep(1);
            if ( WIFEXITED(status) ) printf("Child1: Child 2 exited with exit status %d.\n",
                                                                        WEXITSTATUS(status));
            else printf("Child 1: child has not terminated correctly.\n");
        }

    } else {
         /* then we're the parent process, "Parent" */
        printf("Parent: fork() went ok. My child's PID is %d\n",
        childpid); /* wait for the child to terminate and report about that
        */ wait(&status);
        if ( WIFEXITED(status) ) printf("Parent: child has exited with status %d.\n",
                                                    WEXITSTATUS(status));
        else printf("Parent: child has not terminated normally.\n");
    }
    return 0;
}
```

4.5 PRE-LAB QUESTIONS

1. How to write arithmetic multiplication operator in shell.
2. Write down the syntax for nested if statement.

4.6 LAB ASSIGNMENT

1. Write a shell script to count number of txt,c and shell programs present in current directory.
2. Write a shell script to count number of only files present in current directory.

4.7 POST-LAB QUESTIONS

1. What is means by relation operator , name any three relation operators present in shell.
2. What is meant by logic operator, and Explain about each operator.

EXPERIMENT 5

5.1 OBJECTIVE

Write a shell script that accepts a list of file names as its arguments, counts and reports the occurrence of each word that is present in the first argument file on other argument files.

5.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

5.3 PROGRAM LOGIC

Read list of file names and counts and report the occurrence of each word that is present in the first argument file on other argument files using comm and grep commands.

5.4 DESCRIPTION / PROCEDURE

```
echo Enter file name:
read file1
read file2
a=`comm -2 $file1 $file2`
b=`grep -c $a $file2`
echo Words contained in file one occurred in file two $b times
grep -n $a $file2
```

INPUT: sh prog5.sh

```
Enter file name: f1
myfile
```

OUTPUT:

```
Words contained in file one occured in file two 3 times
1:myfile contains
5:myfile
8:myfile
```

5.5 PRE-LAB QUESTIONS

1. What is difference between comm and diff commands
2. What is the significance of using single and double quotation in echo statement.
3. What are environmental variables.

5.6 LAB ASSIGNMENT

1. Write a script to reverse a given string.
2. Write a script to copy list of file into specified directory.

5.7 POST-LAB QUESTIONS

1. What is difference between grep, egrep and fgrep commands
2. How to copy and paste lines in a vi-editor
3. What are different types of shell and how to move from one shell to other shell

EXPERIMENT 6

6.1 OBJECTIVE

Write a shell script to list all of the directory files in a directory.

6.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

6.3 PROGRAM LOGIC

1. Read a directory
2. Test given directory is directory file and exist using test options -d
3. If its directory and exist display all the sub directories and files to output stream
4. else display its not directory or not exists.

6.4 DESCRIPTION / PROCEDURE

```
echo " Enter dir name "
read dir
if [ -d $dir ]
then
   echo " Files in $dir are
   " ls $dir
else
   echo " Dir does not exist"
fi
```

INPUT:

```
sh Lp6.sh
Enter dir name
Presanna
```

OUTPUT:

```
Files   in   prasanna
are 3.sh
4.sh
pp2.txt
```

6.5 PRE-LAB QUESTIONS

1. Name few test commands present in unix.
2. Write down the syntax case statement

6.6 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.

6.7 POST-LAB QUESTIONS

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

EXPERIMENT 7

7.1 OBJECTIVE
Write a shell script to find factorial of a given number.

7.2 RESOURCE/REQUIREMENTS
Linux operating system ,vi-editor, shell-interpreter

7.3 PROFRAM LOGIC
1. Read a filename
2. take a variable i for count and execute expression in for loop ,increment the variable till loop ends,
3. Display the factorial of given number to output stream.

7.4 DESCRIPTION / PROCEDURE
```
echo Factorial
echo Enter number:
read n
fact=1
i=1
for((i=1;i<=n;i++))
do
   fact=`expr $fact \* $i`
done
echo Factorial of $n is $fact
```
INPUT:
```
sh  p7.sh
Factorial
Enter number:5
```
OUTPUT:
```
Factorial of 5 is 120
```

7.5 PRE-LAB QUESTIONS
1. What is the use of while loop.
2. How you do command substitution in shell script.
3. How do you access command line arguments from within a shell script?

7.6 LAB ASSIGNMENT
1. Write a shell script for menu driven to execute a command like 1 for ls,2 for ls –l, etc..
2. Write a shell script to find the sum of digits.

7.7 POST-LAB QUESTIONS
1. Illustrate the difference between while and for loop.
2. Which operator is used to check string is Null .
3. What is the name of the variable which counts number of arguments passed?

EXPERIMENT 8

8.1 OBJECTIVE

Write an awk script to count the number of lines in a file that do not contain vowels.

8.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

8.3 PROGRAM LOGIC

1. Initialize total=0 in begin part
2. Using if command check each line to count the number of lines in a file that do not contain vowels in body part.
3. Display total lines to output stream that do not contain vowels in end part

8.4 DESCRIPTION / PROCEDURE

```
BEGIN{ print Displaying number of lines in a file that do not contain vowels
        total=0}
{if($0!~/[aeiouAEIOU]/)
total=total + 1}
END{print "The total lines in a file that do not contain vowels:",total}
```

INPUT:

```
awk prog8.awk lp1.sh
Displaying number of lines in a file that do not contain vowels
```

OUTPUT:

```
The total lines in a file that do not contain vowels:1
```

8.5 PRE-LAB QUESTIONS

1. Define awk? What are the features of awk script?
2. What is the syntax for awk script?

8.6 LAB ASSIGNMENT

1. Write a awk script to display between lines eg.2 to 8 lines to output stream.
2. Write a awk script to find sum of total salary of all employees of given file.

8.7 POST-LAB QUESTIONS

1. Illustrate difference between awk and sed.
2. Write a awk command to search given pattern in file if found display to output entire line.

EXPERIMENT 9

9.1 OBJECTIVE

Write an awk script to find the number of characters, words and lines in a file.

9.2 RESOURCE/REQUIREMENTS

Linux operating system ,vi-editor, shell-interpreter

9.3 PROGRAM LOGIC

Write awk script to find the number of characters, words and lines in a file

9.4 DESCRIPTION / PROCEDURE

```
BEGIN{ print Displaying number of characters, words and lines in a
file} {word=words + NF}
{len = length($0)}
{charcount=charcount + len}
END{print The total number of characters, words and lines in a file is:
print("Words:\t",words)
print("Lines:\t",NR)
print("Chars:\t",len) }
```

INPUT:

```
awk prog9.awk lp5.sh
```

OUTPUT:

```
The total number of characters, words and lines in a file
is: Words:12
Lines:3
Chars:39
```

9.5 PRE-LAB QUESTIONS

1. How input file is given to awk script?
2. What is the use of next, getline and exit control actions in awk

9.6 LAB ASSIGNMENT

1. Write a awk script to convert lower case characters to upper case of given file.

9.7 POST-LAB QUESTIONS

1. Explain about associative arrays in awk.

EXPERIMENT 10

10.1 OBJECTIVE

Write a C program that makes a copy of a file using Systems calls

10.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, shell interpreter

10.3 PROGRAM LOGIC

1. Read source filename, destination filename
2. Open given source filename
3. Read the content from file and write to destination file
4. Repeat step 3 till end of file reaches
5. close open files

10.4 DESCRIPTION / PROCEDURE

```
#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);
}
```

I***NPUT:*** 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
```

10.5 PRE-LAB QUESTIONS

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

10.6 LAB ASSIGNMENT

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

10.7 POST-LAB QUESTIONS

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

EXPERIMENT 11

11.A.1 OBJECTIVE

Write a C Program to Implement the Unix command cat using system calls.

11.A.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, shell interpreter

11. A.3 PROGRAM LOGIC

1. Open file which is input given by command line arguments
2. Read content from opened file
3. Display content to output stream
4. Repeat step 2 and step 3 till end of file reach.

11. A.4 DESCRIPTION / PROCEDURE

```
#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

11. A.5 PRE-LAB QUESTIONS

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

11. A.6 LAB ASSIGNMENT

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

11. A.7 POST-LAB QUESTIONS

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

11.B.1 OBJECTIVE

Write a C Program to Implement the Unix command ls using system calls

11. B.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

11. B.3 PROGRAM LOGIC

1. Open a directory of given directory name
2. Scan directory and Read file and display filename to output stream
3. Repeat step 2 till eof directory reach.

11. B.4 DESCRIPTION / PROCEDURE

```
#include <sys/types.h>
#include <sys/dir.h>
#include <sys/param.h>
#include <stdio.h>
#define FALSE 0
#define TRUE 1
extern int alphasort();
char pathname[MAXPATHLEN];
main() {
int count,i;
struct dirent **files;
int file_select();
if (getcwd(pathname) == NULL )
{ printf("Error getting pathn");
exit(0);
}
printf("Current Working Directory = %sn",pathname);
count = scandir(pathname, &files, file_select, alphasort);
if (count <= 0)
{ printf("No files in this directoryn");
exit(0);
}
printf("Number of files = %d
\n",count); for (i=1;i<count+1;++i)
printf("%s \n",files[i-1]->d_name);
}
int file_select(struct direct *entry)
{
if ((strcmp(entry->d_name, ".") == 0) ||(strcmp(entry->d_name, "..") == 0))
return (FALSE);
else
return (TRUE);
 }
```

INPUT:

```
Cc prog11c.c
```

OUTPUT:

```
Current Working Directory =/home/prasanna
Number of files:2
Lp1.sh
 lp2.sh
```

11. B.5 PRE-LAB QUESTIONS

1. What is the difference between lseek() and seekdir();

11. B.6 LAB ASSIGNMENT

1. Write a c-program to reverse a file using lseek();

11. B.7 POST-LAB QUESTIONS

1. What is the difference between system call and library functions

11.C.1 OBJECTIVE

Write a C Program to Implement the Unix command mv using system calls.

11. C.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

11. C.3 PROGRAM LOGIC

1. Open two files. Read and write mode respectively.
2. Using rename function move file content to another file
3. Remove source file using unlink() function.

11. C.4 DESCRIPTION / PROCEDURE

```
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
int main(int argc, char **argv)
{
  int fd1,fd2;
  int n,count=0;
  fd1=open(argv[1],O_RDONLY);
fd2=creat(argv[2],S_IWUSR);
rename(fd1,fd2);
unlink(argv[1]);
return (0);
}
```

INPUT:

```
cc mv.c file1 file2
```

OUTPUT:

```
# creates file2 and copies the content of file1 to file2 and removes file1
```

11. C.5 PRE-LAB QUESTIONS

1. What is the difference between cp and mv commands?

11. C.6 LAB ASSIGNMENT

1. Write a C program to move file content to another file without using functions rename and unlink.
2. Write a C program to copy content from source to destination file

11. C.7 POST-LAB QUESTIONS

1. What is difference between file descriptor and file pointer.

EXPERIMENT 12

12.1 OBJECTIVE

Write a C program that takes one or more file or directory names as command line input and reports the following information on the file.

1. file type
2. number of links
3. read, write and execute permissions
4. time of last access

(Note: use /fstat system calls)

12.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

12.3 PROGRAM LOGIC

1. Open a file using fopen() function
2. Read a file and display a file properties to output stream.

12.4 DESCRIPTION / PROCEDURE

```
#include<stdio.h>
main()
{
FILE *stream;
int buffer_character;
stream=fopen("test","r");
if(stream==(FILE*)0)
{
fprintf(stderr,"Error opening file(printed to standard error)\n");
fclose(stream);
exit(1);
}}
if(fclose(stream))==EOF)
{
fprintf(stderr,"Error    closing    stream.(printed    to    standard
error)\n); exit(1);
}
return();
}
```

12.5 PRE-LAB QUESTIONS

1. What is the difference between stat( ), fstat() and lstat() functions

12.6 LAB ASSIGNMENT

1.Write a C program to count number of words, lines and characters using system calls

12.7 POST-LAB QUESTIONS

1.List properties of files and different types of files in Linux

EXPERIMENT 13

13.1 OBJECTIVE

Write a C program to emulate the Unix ls – l command**.**

13.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

13.3 PROGRAM LOGIC

1. Declare and initialize required objects.
2. Read the directory name form the user.
3. Open the directory using opendir() system call and report error if the directory is not available.
4. Read the entry available in the directory.
5. Display the directory entry ie., name of the file or sub directory.
6. Repeat the step 6 and 7 until all the entries were read.

13.4 DESCRIPTION / PROCEDURE

```
#include <stddef.h>
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h> int
main (void) {
DIR *dp;
struct dirent *ep;
dp = opendir (".");
if (dp != NULL) {
while (ep = readdir (dp))
printf("%s\n", ep->d_name);
closedir (dp);
}
else
perror ("Couldn't open the directory");
return 0;
}
```

```
INPUT:  cc 13.c
 ./a.out
OUTPUT: 2  3  a.out  1  4 …. . 13.c
```

13.5 PRE-LAB QUESTIONS

1. What is the purpose of 0_CREAT and O_SYNC.

13.6 LAB ASSIGNMENT

1. Write a C-program to simulate `nl` command.

13.7 POST-LAB QUESTIONS

1. What is the use of -> operator?

EXPERIMENT 14

14.1 OBJECTIVE

Write a C program to list for every file in a directory, its inode number and file name

14.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

14.3 PROGRAM LOGIC

1. Read Directory name as input
2. Scan entire directory and display its directory file name and inode number till eof of the directory

14.4 DESCRIPTION / PROCEDURE

```
#include<stddef.h>
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>
main()
{
DIR *dp; Struct
dirent *p;
char dname[20];
Struct stat x;
Printf("Enter the directory
name:"); Scanf("%s",dname);
Dp=opendir(dname);
Printf("\n FILE NAME \t INODE NUMBER
\n"); While((p=readdir(dp))!=NULL)
{
        Printf("%s\t %/d\n",p->d_name,x.st_ino);
}
}
```

INPUT:

```
 cc inode.c –o inode
./inode
```

OUTPUT:

```
  FILE NAME   INODE NUMBER
…………..       4195164
File2.c       4195164
….
File1.c       4195164
```

14.5 PRE-LAB QUESTIONS

1. What is the system call that change the directory.
2. List all directory systems calls with example.

14.6 LAB ASSIGNMENT

1.Write a C-program to display all files using system calls

14.7 POST-LAB QUESTIONS

1.What flag is used to append the data /value in the file.

EXPERIMENT 15

15.1 OBJECTIVE

Write a C program that demonstrates redirection of standard output to a file. Ex: ls >f1.
/ freopen example: redirecting stdout */*

15.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

15.3 PROGRAM LOGIC

1. Open file using freopen() function
2. Redirect of standard to out put to a new file

15.4 DESCRIPTION / PROCEDURE

```
#include  <stdio.h>
int main ()
{
freopen ("myfile.txt","w",stdout);
printf ("This sentence is redirected to a file which is given at
output."); fclose (stdout);
return 0;
}
```

INPUT:

```
cc 15.c –o file1
./file1
```

OUTPUT:

```
This sentence is redirected to a file which is given at output
```

15.5 PRE-LAB QUESTIONS

1.What is the purpose of fopen(),fread(),fwrite() functions

15.6 LAB ASSIGNMENT

1.Write a C-program to append content of file to another file

15.7 POST-LAB QUESTIONS

1. Explain different streams in linux
2. Illustrate different types of files in linux

EXPERIMENT 16

16.1 OBJECTIVE

Write a C program to create a child process and allow the parent to display “parent” and the child to display “child” on the screen.

16.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

16.3 PROGRAM LOGIC

1. Create child process using fork() function
2. If pid is not equal to zero execute parent block
3. else if pid is equal to zero execute child block
4. Display process id using getpid() and parent process id getppid() functions.

16.4 DESCRIPTION / PROCEDURE

```
#include
<stdio.h> main()
{
   int pid;
   printf("I'm the original process with PID %d and PPID %d.\n",getpid(),getppid());
   pid=fork(); /* Duplicate. Child and parent continue from here.*/
   if (pid!=0)  {/* pid is non-zero, so I must be the parent  */
         printf("I'm the parent process with PID %d and PPID %d.\n",getpid(),getppid());
         printf("My child's PID is %d.\n", pid);
      }
   else { /* pid is zero, so I must be the child. */
         printf("I'm the child process with PID %d and PPID %d.\n",getpid(),getppid());
      }
   printf("PID %d terminates.\n",pid); /* Both processes execute this */
}
```

INPUT:

```
$cc fork.c              ... run the program.
./a.out
```

OUTPUT:

```
I'm the original process with PID 13292 and PPID 13273.
I'm the parent process with PID 13292 and PPID 13273.
My child's PID is 13293.
I'm the child process with PID 13293 and PPID 13292.
PID 13293 terminates.   ... child terminates.
PID 13292 terminates.   ... parent terminates.
```

16.5 PRE-LAB QUESTIONS

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

16.6 LAB ASSIGNMENT

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

16.7 POST-LAB QUESTIONS

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

EXPERIMENT 17

17.1 OBJECTIVE

Write a C program to create a Zombie process.

17.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

17.3 PROGRAM LOGIC

1. Create a new child process using fork()
2. If pid is not equal to zero execute parent process and block the process 1000 seconds
3. If pid is equal to zero executer child process and terminate immediately
4. To see already terminated process i.e zombie process open new command prompt and type ps -a it shows status **z** means that process is zombie process.

17.4 DESCRIPTION / PROCEDURE

```
#include
<stdio.h> main(){
   int pid;
   pid=fork(); /* Duplicate */
   if (pid!=0) /* Branch based on return value from fork() */      {
        while (1) /* never terminate, and never execute a wait() */
           sleep(1000);      }
   else      {
        exit(42); /* Exit with a silly number */
      }}
```

INPUT:

```
$ cc prog17.c
./a.out& ... execute the program in the background.
[1] 13545
```

OUTPUT:

```
$ ps
  PID TT STAT TIME COMMAND
13535 p2 s    0:00 -ksh(ksh) ... the shell
13545 p2 s    0:00 aombie.exe...   the parent process
13536 p2 z    0:00 <defunct> ...  the zombie child process
13537 p2 R    0:00 ps
$ kill 13545                ... kill the parent process.
[1] Terminated  zombie.exe
$ ps                        ... notice the zombie is gone now.
   PID TT STAT TIME COMMAND
13535 p2 s    0:00 -csh(csh)
13548 p2 R    0:00 ps
```

17.5 PRE-LAB QUESTIONS

1. Define Zombie process.
2. How to know the status of process using commands.

17.6 LAB ASSIGNMENT

1. Write a C-program to create a child process to convert lower to upper case letters.

17.7 POST-LAB QUESTIONS

1. Explain different ways to terminate the process?

EXPERIMENT 18

18.1 OBJECTIVE

Write a C program that illustrates how an orphan is created.

18.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

18.3 PROGRAM LOGIC

1. Create a new child process using fork()
2. If pid is not equal to zero execute parent process block of code and terminates
3. If pid is equal to zero executer child process and block process for 5 seconds in mean time parent process terminates.
4. To see child process is became orphan process open new command prompt and type
   ps -a it shows status **O** means that process is orphan process.

18.4 DESCRIPTION / PROCEDURE

```
#include <stdio.h>
main()
{
   int pid;
   printf("I'm the original process with PID %d and PPID
         %d.\n", getpid(),getppid());
   pid=fork(); /* Duplicate. Child and parent continue from here.*/
   if (pid!=0) /* Branch based on return value from fork() */
     { /* pid is non-zero, so I must be the parent */
        printf("I'm the parent process with PID %d and PPID
              %d.\n", getpid(),getppid());
        printf("My child's PID is %d.\n", pid);
     }
   else { /* pid is zero, so I must be the child. */
        sleep(5); /*Make sure that the parent terminates first. */
        printf("I'm the child process with PID %d and PPID %d.\n",
             getpid(),getppid());
     }
   printf("PID %d terminates.\n",pid); /* Both processes execute this */}
```

INPUT:

```
$cc prog18.c
./a.out              ... run the program.
```

OUTPUT:

```
I'm the original process with PID 13364 and PPID 13346.
I'm the parent process with PID 13364 and PPID 13346.
PID 13364 terminates.
I'm the child process with PID 13365 and PPID 1.  ...orphaned!
PID 13365 terminates.   ... child terminates.
```

18.5 PRE-LAB QUESTIONS

1.Illustrate difference between zombie process and orphan process.

18.6 LAB ASSIGNMENT

1. Write a C program to find given number is Armstrong number or not by parent process.

18.7 POST-LAB QUESTIONS

1. What is the child parent process id if child process status is orphan process?

EXPERIMENT 19

19.1 OBJECTIVE

Write a C program that illustrates how to execute two commands concurrently with a command pipe. Eg. ls-l|sort.

19.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

19.3 PROGRAM LOGIC

1. Open a directory of given directory name
2. Scan directory and Read file and display filename to output stream
3. Repeat step 2 till eof directory reach.
4. Using stat function test type of file, its permissions and file properties and display to output stream.

19.4 DESCRIPTION / PROCEDURE

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
#include<dirent.h>
main(int argc,char *argv[])
{
    struct dirent *p;
    DIR *dp; struct
    stat b;
    dp=opendir(".");
    if(dp==NULL)
    {
        printf("\nDirectory opening problem\n");
        return;
    }
    while((p=readdir(dp))!=NULL)
    {
        if(stat(p->d_name,&b)<0)
        {
            printf("\nStat failure \n");
            exit(0);
        }
      switch(b.st_mode & S_IFMT )
       {
            case S_IFREG: printf(" -
                    "); break;
            case S_IFDIR: printf(" d");
```

```
                                    break;
                        case S_IFCHR: printf(" c");
                                    break;
                        case S_IFBLK: printf(" b");
                                    break;
                        case S_IFLNK: printf(" l");
                                    break;
                        case S_IFSOCK: printf("
                                    s"); break;
                        case S_IFIFO: printf(" p");
                                    break;
                  }
                  if(S_IRUSR & b.st_mode)
                        printf(" r");
                  else
         printf(" -"); if(S_IWUSR
       & b.st_mode) printf(" w");

                  else
                        printf(" -");
                  if(S_IXUSR & b.st_mode)
                        printf(" x");
                  else
                        printf(" -");
                  if(S_IRGRP & b.st_mode)
                        printf(" r");
                  else
                        printf(" -");
                  if(S_IWGRP & b.st_mode)
                        printf(" w");
                  else
                        printf(" -");
                  if(S_IXGRP & b.st_mode)
                        printf(" x");
                  else
                        printf(" -");
                  if(S_IROTH & b.st_mode)
                        printf(" r");
                  else
      printf(" -");
      if(S_IWOTH & b.st_mode)
                        printf(" w");
                  else
                        printf(" -");
                  if(S_IXOTH & b.st_mode)
                        printf(" x");
                  else
                        printf(" -");
                  printf("%3d ",b.st_nlink);
                  printf("%4d ",b.st_uid);
                  printf("%4d ",b.st_gid);
                  printf("%6d ",b.st_size);
                  printf("%9ld",b.st_ctime);
```

```
                printf("  %s\n",p->d_name);
        }
}
```

INPUT:

```
vi 19.c
cc 19.c
 ./a.out
```

OUTPUT:

```
 - r w - r w - r - - 1 500 500      1506 1380610351 19.c
 - r w - r w - r - - 1 500 500         0   1380523478   2
 - r w - r w - r - - 1 500 500         0   1380523478   3
 - r w x r w x r - x 1 500 500      6038 1380610357 a.out
 - r w - r w - r - - 1 500 500         0   1380523478   1
 - r w - r w - r - - 1 500 500       421 1380524812 12.c
 - r w - r w - r - - 1 500 500         0   1380523478   4
 d r w x - - - - - - 15 500 500     4096 1380609957 ..
 d r w x r w x r - x 2 500 500      4096 1380610357 .
- r w - r w - r - - 1 500 500       347 1380523684 13.c
```

19.5 PRE-LAB QUESTIONS

1. What is the purpose of stat() functions.
2. Define file system?

19.6 LAB ASSIGNMENT

1. Write a C program to find number of sub directories in given directory.

19.7 POST-LAB QUESTIONS

1. Which system call is used to remove any type of file.

EXPERIMENT 20

20.1 OBJECTIVE
Write a C program in which a parent writes a message to a pipe and the child reads the message.

20.2 RESOURCE/REQUIREMENTS
Linux operating system, vi –editor, c-compiler

20.3 PROGRAM LOGIC
1. Create pipe using pipe() function and new process using fork() function
2. If parent process executing write a message to pipe at one end
3. If child process executing read a message from pipe at other end which is written by parent

20.4 DESCRIPTION / PROCEDURE
```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
main() {
    int pfds[2];
    char buf[30];
   if(pipe(pfds) == -1)     {
        perror("Pipe is not created");
        exit(1);     }
    if(!fork())     {
        printf("CHILD: Writing to the pipe\n");
        write(pfds[1],"Hello World, I am child",23);
        printf("CHILD:Exiting");
        exit(0);     }
    else     {
        printf("PARENT:reading from the pipe\n");
        read(pfds[0],buf,23);
        printf("PARENT:Received Data is :%s",buf);
    }}
```
INPUT: cc
prog20.c
./a.out

OUTPUT:
```
CHILD: Writing to the pipe
CHILD:Exiting
PARENT:reading from the pipe
PARENT:Received Data is : Hello World, I am child
```

20.5 PRE-LAB QUESTIONS
1. What are the return values of pipe function?

20.6 LAB ASSIGNMENT
1. Implement two way communication using pipes.

20.7 POST-LAB QUESTIONS
1. What is the use of read and write system calls
2. How many processes will create if 4 fork functions executed?
3. Illustrate difference between pipes and named pipes?

EXPERIMENT 21

21.1 OBJECTIVE

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.

21.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

21.3 PROGRAM LOGIC

1. Create process using msgget(), identify process using a key
2. Declare structure to store message and message type.
3. Using msgsnd() function write a message to message queue.

21.4 DESCRIPTION / PROCEDURE

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
main()
{
int qid,len,i;
char s[15];
struct
{
long mtype;
char mtext[15];
}message,buff;
qid=msgget((key_t)10,IPC_CREAT|0666);
if(qid==-1)
{
perror("message queue create failed");
exit(1);
}
for(i=1;i<=3;i++)
{
printf("Enter the message to send\n");
scanf("%s",s); strcpy(message.mtext,s);
message.mtype=i;
len=strlen(message.mtext);
if(msgsnd(qid,&message,len+1,0)==-1)

{
perror("message failed\n");
exit(1);
}
}
}
```

INPUT: cc
prog21.c

OUTPUT:

```
Enter the message to send: hi
Enter the message to send: hello, how are you
Enter the message to send: bye
```

21.5 PRE-LAB QUESTIONS

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

21.6 LAB ASSIGNMENT

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

21.7 POST-LAB QUESTIONS

1. Describe use of pipe, fifos and message queues

EXPERIMENT 22

22.1 OBJECTIVE

Write a C program (receiver.c) that receives the messages (from the above message queue as specified in (22)) and displays them.

22.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

22.3

i) Connect client and server with socket() fuction.
ii) Provide socket address (port,Ip-address).
iii) Use connect() and bind(),listen() in client and server respectively.
iv) If successful use accept() in server.
v) Use fprintf() and read() for sending and receiving the data for communication between server and client.
vi) Close the connectrion

22.4 DESCRIPTION / PROCEDURE

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
main()
{
int qid,len,i;
char s[15];
struct
{
long mtype;
char mtext[15];
}buff;
qid=msgget((key_t)10,IPC_CREAT|0666);
if(qid==-1)
{
perror("message queue create failed");
exit(0);
}
for(i=1;i<=3;i++)
{
if(msgrcv(qid,&buff,15,i,0)==-1)
{
perror("message failed\n");
exit(0);
}
printf("Message received from sender is %s\n",buff.mtext);
}
}
```

INPUT:

```
 cc prog22.c
./a.out
```

OUTPUT:

```
Message received from sender is: hi
Message received from sender is: hello, how are you
Message received from sender is: bye
```

22.5 PRE-LAB QUESTIONS

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

22.6 LAB ASSIGNMENT

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

22.7 POST-LAB QUESTIONS

1. Which system call is used to remove or delete a message queue

EXPERIMENT 23

23.1 OBJECTIVE

Write a C program to allow cooperating processes to lock a resource for exclusive use, using
a) Semaphores b) flock or lockf system calls.

23.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

23.3 PROGRAM LOGIC

1. Create semaphore and create key using ftok() function
2. Use semctl() to control the process.

23.4 DESCRIPTION / PROCEDURE

```
#include<stdio.h>
#include<stdlib.h>
#include<error.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
int main(void)
{
key_t  key;
int semid;
union semun arg;
if((key==ftok("sem demo.c","j"))== -1)
{
perror("ftok");
exit(1);
}
if(semid=semget(key,1,0666|IPC_CREAT))== -1)
{
perror("semget"):
exit(1);
}
arg.val=1;
if(semctl(semid,0,SETVAL,arg)== -1)
{
perror("smctl");
exit(1);
}
return 0;
}
```

23.5 PRE-LAB QUESTIONS

1. How to control the process in shared segment

23.6 LAB ASSIGNMENT

1. **Write** a program for shared memory form of IPC using producer consumer relation in such a way that consumer should read only after the producer has written some text to the shared memory.

23.7 POST-LAB QUESTIONS

1. Which system call is used to remove or delete a semaphore?

EXPERIMENT 24

24.1 OBJECTIVE

Write a C program that illustrates suspending and resuming processes using signals.

24.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

24.3 PROGRAM LOGIC

1. Use kill() to suspend and resume process

24.4 DESCRIPTION / PROCEDURE

```
#include<sys/types.h>
#include<signal.h>
//suspend the process(same as hitting
crtl+z) kill(pid,SIGSTOP);

//continue the process
kill(pid,SIGCONT);
```

24.5 PRE-LAB QUESTIONS

1. Define signal? Explain how to handle signals.

24.6 LAB ASSIGNMENT

1. Write a program to handle the signals like SIGINT , SIGFPE. SIGQUIT

24.7 POST-LAB QUESTIONS

1. Which signals cannot catch using signal handler functions.

EXPERIMENT 25

25.1 OBJECTIVE

Write a C program that implements a producer-consumer system with two processes. (using Semaphores).

25.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

25.3 PROGRAM LOGIC

1. create semaphore using semget( ) system call
2. if successful it returns positive value
3. create two new processes
4. first process will produce
5. until first process produces second process cannot consume

25.4 DESCRIPTION / PROCEDURE

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
#include<unistd.h>
#define num_loops 2
int main(int argc,char* argv[])
{
int sem_set_id;
int child_pid,i,sem_val;
struct sembuf sem_op;
int rc;
struct timespec delay;
clrscr();
sem_set_id=semget(ipc_private,2,0600);
if(sem_set_id==-1)
{
perror("main:semget");
exit(1);
}
printf("semaphore set created,semaphore setid,,%d‟\n
", sem_set_id);
child_pid=fork();
switch(child_pid)
{
case -1:
perror("fork");
exit(1);
case 0:
for(i=0;i<num_loops;i++)
{
sem_op.sem_num=0;
sem_op.sem_op=-1;
sem_op.sem_flg=0;
semop(sem_set_id,&sem_op,1);
printf("producer:‟%d‟\n",i);
fflush(stdout);
}
break;
default:
for(i=0;i<num_loops;i++)
{
printf("consumer:‟%d‟\n",i);
fflush(stdout);
sem_op.sem_num=0;
sem_op.sem_op=1;
sem_op.sem_flg=0;
semop(sem_set_id,&sem_op,1);
if(rand()>3*(rano_max14));
{
delay.tv_sec=0;
delay.tv_nsec=10;
nanosleep(&delay,null);
}
}
break;
}
return 0;
}
```

INPUT AND OUTPUT:

```
semaphore set created
semaphore set id
„327690‟ producer: „0‟
consumer:‟0‟
producer:‟1‟
   consumer:‟1‟
```

25.5 PRE-LAB QUESTIONS

1. Define semaphores? What is their purpose? List and explain the APIs used to create and control the semaphores.

25.6 LAB ASSIGNMENT

1. Write a C program to remove semaphores using semaphore functions.

25.7 POST-LAB QUESTIONS

1. Which system call is used to remove or delete a semaphores?

EXPERIMENT 26

26.1 OBJECTIVE

Write client and server programs (using c) for interaction between server and client processes using Unix Domain sockets.

26.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

26.3 PROGRAM LOGIC

1. Create socket at client side and server side for communication.
2. Server side execute bind and accept function to accept client
3. Client side execute connect function to establish connection between client and server
4. Exchange information through unix domain sockets.
5. Close socket file descriptor after completion of communication.

26.4 DESCRIPTION / PROCEDURE

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

int connection_handler(int connection_fd)
{
 int nbytes;
 char buffer[256];

 nbytes = read(connection_fd, buffer,
 256); buffer[nbytes] = 0;

 printf("MESSAGE FROM CLIENT: %s\n", buffer);
 nbytes = snprintf(buffer, 256, "hello from the
 server"); write(connection_fd, buffer, nbytes);

 close(connection_fd);
 return 0;
}

int main(void)
{
 struct sockaddr_un address;
 int socket_fd, connection_fd;
 socklen_t address_length;
 pid_t child;

 socket_fd = socket(PF_UNIX, SOCK_STREAM,
 0); if(socket_fd < 0)
 {
  printf("socket()
  failed\n"); return 1;
 }

 unlink("./demo_socket");

 /* start with a clean address structure */
 memset(&address, 0, sizeof(struct sockaddr_un));

 address.sun_family = AF_UNIX;
 snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

 if(bind(socket_fd,
        (struct sockaddr *) &address,
        sizeof(struct sockaddr_un)) != 0)
 {
  printf("bind()
  failed\n"); return 1;
 }

 if(listen(socket_fd, 5) != 0)
 {
  printf("listen()
  failed\n"); return 1;
 }

 while((connection_fd = accept(socket_fd, (struct
                              sockaddr *) &address,
                              &address_length)) > -1)
 {
  child = fork();
  if(child == 0)
  {
   /* now inside newly created connection handling process
   */ return connection_handler(connection_fd);
  }

  /* still inside server process
  */ close(connection_fd);
 }

 close(socket_fd);
 unlink("./demo_socket");
 return 0;
}
```

Client.c

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

int main(void)
{
```

```
struct sockaddr_un address;
int socket_fd, nbytes;
char buffer[256];

socket_fd = socket(PF_UNIX, SOCK_STREAM,
0); if(socket_fd < 0)
{
 printf("socket()
 failed\n"); return 1;
}

/* start with a clean address structure */
memset(&address, 0, sizeof(struct sockaddr_un));

address.sun_family = AF_UNIX;
snprintf(address.sun_path, UNIX_PATH_MAX, "./demo_socket");

if(connect(socket_fd,
          (struct sockaddr *) &address,
          sizeof(struct sockaddr_un)) != 0)
{
 printf("connect() failed\n");
 return 1;
}

nbytes = snprintf(buffer, 256, "hello from a
client"); write(socket_fd, buffer, nbytes);

nbytes = read(socket_fd, buffer, 256);
buffer[nbytes] = 0;
printf("MESSAGE FROM SERVER: %s\n",
buffer); close(socket_fd);
return 0;
}
```

26.5 PRE-LAB QUESTIONS

1. Define socket?
2. Differentiate between connection oriented and connection less communication.

26.6 LAB ASSIGNMENT

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.

26.7 POST-LAB QUESTIONS

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

EXPERIMENT 27

27.1 OBJECTIVE

Write client and server programs (using c) for interaction between server and client processes using Internet Domain sockets.

27.2 RESOURCE/REQUIREMENTS

Linux operating system, vi –editor, c-compiler

27.3 PROGRAM LOGIC

1. Connect client and server with socket() fuction.
2. Provide socket address (port, Ip-address) of Internet Domain Sockets.
3. Use connect() and bind(),listen() in client and server respectively.
4. If successful use accept() in server.
5. Use fprintf() and read() for sending and receiving the data for communication between server and client.
6. Close the connection

27.4 DESCRIPTION / PROCEDURE

Server.c

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

int main(int argc, char *argv[])
{
   int listenfd = 0, connfd = 0;
   struct sockaddr_in serv_addr;

   char sendBuff[1025];
   time_t ticks;

   listenfd = socket(AF_INET, SOCK_STREAM,
   0); memset(&serv_addr, '0', sizeof(serv_addr));
   memset(sendBuff, '0', sizeof(sendBuff));

   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
   serv_addr.sin_port = htons(5000);

   bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
```

```
    listen(listenfd, 10);

    while(1)
    {
        connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

        ticks = time(NULL);
        snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n",
        ctime(&ticks)); write(connfd, sendBuff, strlen(sendBuff));

        close(connfd);
        sleep(1);
     }
}
```

Client.c

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

int main(int argc, char *argv[])
{
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if(argc != 2)
    {
        printf("\n Usage: %s <ip of server> \n",argv[0]);
        return 1;
    }

    memset(recvBuff, '0',sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket
        \n"); return 1;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);

    if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
    {
        printf("\n inet_pton error
        occured\n"); return 1;
    }

    if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
       printf("\n Error : Connect Failed
       \n"); return 1;
    }

    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
        recvBuff[n] = 0; if(fputs(recvBuff,
        stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }

    if(n < 0)
    {
        printf("\n Read error \n");
    }

    return 0;
}
```

27.5 PRE-LAB QUESTIONS

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

27.6 LAB ASSIGNMENT

1. Write a program to implement UDP client server application in which client takes an file name from the command line and sends to the server. Server returns the content of received file to the client.

27.7 POST-LAB QUESTIONS

1. Which system call is used to UDP Client Server Communication.

EXPERIMENT 28

28.1 OBJECTIVE
Implement shared memory form of IPC

28.2 RESOURCE/REQUIREMENTS
Linux operating system, vi –editor, c-compiler

28.3 PROGRAM LOGIC
i) Create two process ,
ii) Use shmget() to connect the processes.
iii) Take a memory and attach this two process using shmat()
iv) Now both the process can access the common memory.

28.4 DESCRIPTION / PROCEDURE

```
#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 Linux operating system, vi –editor, c-compiler= '*')
      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);
    }
```

28.5 PRE-LAB QUESTIONS

1. What are the advantages of using shared memory?
2. shmget() has how many parameters.

28.6 LAB ASSIGNMENT

1. Write a c-program to implement shred memory among more than one process with effecting critical region (i.e. synchronization).

28.7 POST-LAB QUESTIONS

1. Which system call used to assign or allocate memory to the process?