1. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEMS (OPENDIR, READDIR, CLOSEDIR, ETC)

```
#include<stdio.h>
#include<dirent.h>
struct dirent *dptr;
int main(int argc, char *argv[])
{
char buff[100];
DIR *dirp;
printf("\n\n ENTER DIRECTORY NAME");
scanf("%s", buff);
if((dirp=opendir(buff))==NULL)
{
printf("The given directory does not exist");
exit(1);
}
while(dptr=readdir(dirp))
{
printf("%s\n",dptr->d_name);
}
closedir(dirp);
}
```

SAMPLE OUTPUT:

```
Cc opdir.c
./a.out

ENTER THE DIRECTORY NAME  UNIT

CHAP1.C
CHAP2.C
```

PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEM (Fork, Getpid, Exit, Etc)

```
#include<stdio.h>
#include<unistd.h>
main()
{
int pid,pid1,pid2;
pid=fork();
if(pid==-1)
{
printf("ERROR IN PROCESS CREATION \n");
exit(1);
}
if(pid!=0)
{
pid1=getpid();
printf("\n the parent process ID is %d\n", pid1);
}
else
{
pid2=getpid();
printf("\n the child process ID is %d\n", pid2);
}
}
```

SAMPLE OUTPUT:

```
Cc fork.c
./a.out


THE CHILD PROCESS ID IS 8640
THE PARENT PROCESS ID IS 8644
```

2. SIMULATION OF LS & GREP COMMENDS

PROGRAM FOR SIMULATION OF LS UNIX COMMANDS

```
#include<stdio.h>
#include<dirent.h>
main(int argc, char **argv)
{
DIR *dp;
struct dirent *link;
dp=opendir(argv[1]);
printf("\n contents of the directory %s are \n", argv[1]);
while((link=readdir(dp))!=0)
printf("%s",link->d_name);
closedir(dp);
}
```

SAMPLE OUTPUT:

```
 Cc list.c
./a.out os
CONTENTS OF THE DIRECTORY OS ARE
Priority.c
Robin.c
copy
```

PROGRAM FOR SIMULATION OF GREP UNIX COMMANDS

```
#include<stdio.h>
#include<string.h>
#define max 1024
void usage()
{
printf("usage:\t. /a.out filename word \n ");
}
int main(int argc, char *argv[])
```

```
{
FILE *fp;
char fline[max];
char *newline;
int count=0;
int occurrences=0;
if(argc!=3)
{
usage();
exit(1);
}
if(!(fp=fopen(argv[1],"r")))
{
printf("grep: couldnot open file : %s \n",argv[1]);
exit(1);
}
while(fgets(fline,max,fp)!=NULL)
{
count++;
if(newline=strchr(fline, '\n'))
*newline='\0';
if(strstr(fline,argv[2])!=NULL)
{
printf("%s: %d %s \n", argv[1],count, fline);
occurrences++;
}
}
}
```

SAMPLE OUTPUT

```
CAT>SAMP
ONE
ONE TWO
THREE FOUR
Cc grep.c
./a.out samp one

Samp:1  one
Samp:2  one two
```

3

Write programs using the I/O system calls of UNIX operating system (open, read, write, etc)

OPEN SYSTEM CALL

PROGRAM:

```
#include<stdio.h>
int main()
{
int fd;
if((fd=open("file.dat"))==-1)
{
perror("cannot open the file.dat");
exit(0);
}
else
printf("\n FILE OPENED SUCCESSSFULLY");
return 0;
}
```

OUTPUT:

FILE OPENED SUCCESSSFULLY

READ SYSTEM CALL

PROGRAM:

```
#include<stdio.h>
main()
{
char b[20];
int fd,xr;
```

```
if((fd=open(“write”,0))==-1)
{
printf(“cannot open file”);
exit(1);
}
do
{
xr=read(fd,b,20);
b[xr]=’\0’;
printf(“%s”,b);
}
while(xr==20);
close(fd);
}
```

OUTPUT:

balaji.S
II YEAR

WRITE SYSTEM CALL

PROGRAM:

```
#include<stdio.h>
main(int ac,char*av[])
{
int fd;
int i=1;
char*sep=” “;
if(ac<1)
{
printf(“\n INSUFFICIENT ARGUMENTS");
exit(1);
}
if((fd=open(“balaji”,0660))==-1)
{
```

```
printf("\n CANNOT CREATE THE FILE");
exit(1);
}
while(i<ac)
{
write(fd,av[i],(unsigned)strlen(av[i]));
write(fd,sep,(unsigned)strlen(sep));
i++;
}
close(fd);
}
```

OUTPUT:

```
cc write.c
./a.out balaji
cat balaji


os lab
balaji
```