LAB-10 17/1/23

1) Write a C/C++ program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close API's in your program.

```
#include<sys/types.h>
#include<sys/stat.h>
#include<string.h>
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
  char buf[100];
   int fd,n;
  mkfifo(argv[1], S_IFIFO|0777);
  if(argc==3)
  {
  fd = open(argv[1], O_WRONLY);
   write(fd, argv[2], strlen(argv[2]));
    close(fd);
  }
  if(argc==2)
  {
  fd=open(argv[1], O_RDONLY);
    n = read(fd, buf, sizeof(buf));
    buf[n]='\0';
    printf("%s", buf);
     close(fd);
  }
}
```

Output:

```
cc fifo.c
./a.out fifo "5b Linux Lab Prog" &
[1] 3978

./a.out fifo
5b Linux Lab Prog
[1]+ Done ./a.out fifo "5b Linux lab Prog"
```

2) Write a C/C++ program to emulate the unix ln command:

```
#include <unistd.h>
#include <stdio.h>
#include <string.h>
int main (int argc, char *argv[])
{
  if (argc < 3 || argc > 4)
  {
    printf ("Error in usage\n");
    return -1;
  }
  if (argc==4 && strcmp (argv[1], "-s")!=0)
  {
    printf ("for symbolic link use -s option");
    return -1;
  }
  if (argc==4 && access (argv[2], F_OK)==-1)
  {
    printf ("source file does not exist");
    return -1;
  }
  if (argc==3 && access (argv[1], F_OK)==-1)
  {
    printf ("Source file does not exist");
    return -1;
  }
  if (argc==4)
  {
    symlink (argv[2], argv[3]);
    printf ("Symbolic link is created");
    return 0;
  }
  if (argc==3)
  {
    link (argv[1], argv[2]);
    printf ("Hardlink is created");
    return 0;
  }
```

Output:

```
cc link.c
./a.out -s forex.sh slink
Symbolic link is created
```

./a.out file2.sh hardlink
Hardlink is created.

3) Write a C/C++ POSIX complaint program that prints the POSIX defined configuration options

```
#define _POSIX_SOURCE
#define _POSIX_C_SOURCE 199309L
#include <stdio.h>
#include <unistd.h>
int main()
{
#ifdef _POSIX_JOB_CONTROL
printf("system supports Job control feature\n");
#else
printf("system does not support Job control\n");
#endif
#ifdef _POSIX_SAVED_IDS
printf("system supports saved set-UID and
                saved set-GID\n");
#else
printf("system does not support saved set-UID\n");
#endif
#ifdef _POSIX_CHOWN_RESTRICTED
printf("system supports change ownership
                feature\n");

#else
printf("system doesnot support chang
                ownership feature\n");

#endif
#ifdef _POSIX_NO_TRUNC
printf("system supports Path truncation
                option\n");

#else
printf("system does not support Path
                truncation\n");

endif
#ifdef _POSIX_VDISABLE
printf("system supports Disable characters
                for files\n");
#else
printf("system does not support
                disable character\n");
#endif
return 0; }
```

Output:

```
cc posix.c
./a.out
```

System support job control feature
System support saved set UID and saved set -GID

System supports change Ownership feature
System supports Path truncation option
System Supports Disable characters for files

4) Write a C/C++ program to that outputs the contents of its environment list

```
#include<stdio.h>
#include<unistd.h>
int main(int argc, char *argv[])
{
  char **ptr;
  extern char **environ;
  for(ptr= environ ; *ptr; ptr++)
      printf("%s\n", *ptr);
       return 0;
}
```

or

```
int main (int argc, char *argv[], char
                    *envp[])
{
  int i;
  for(i=0; envp[i]!=NULL; i++)
      printf("\n%s", envp[i]);
       getchar();
       return 0;
}
```

Output:

```
cc env.c
% a.out file22.txt
```