# LAB-2

1. **Getcwd:**

   **#include <unistd.h>**

   **char *getcwd(char *buf, size_t size);**

   This function gives the absolute pathname that is the current working directory of the calling process.

   Pathname is returned as the function result and via the argument buf, if present.

2. **Opendir:**

   **#include <sys/types.h>**
   **#include <dirent.h>**

   **DIR *opendir(const char *name);**

   It opens a directory stream corresponding to the directory name, and returns a pointer to the directory stream.

   The stream is positioned at the first entry in the directory.

   On error, NULL is returned, and errno is set appropriately.

3. **Readdir:**

   **#include <dirent.h>**

   **struct dirent *readdir(DIR *dirp);**

   It returns a pointer to a dirent structure representing the next directory entry in the directory stream pointed to by dirp.

   It returns NULL on reaching the end of the directory stream.

   On Linux, the dirent structure is defined as follows:

```
struct dirent {
   ino_t          d_ino;                    /* inode number */
   off_t          d_off;                    /* offset to the next dirent */
   unsigned short d_reclen;       /* length of this record */
   unsigned char  d_type;                   /* type of file*/
   char           d_name[256];              /* filename */
};
```

   On success, readdir() returns a pointer to a dirent structure.

4. **Closedir:**

**#include <sys/types.h>**
**#include <dirent.h>**

**int closedir(DIR *dirp);**

The closedir() function closes the directory stream associated with dirp.

A successful call to closedir() also closes the underlying file descriptor associated withdirp.

The directory stream descriptor dirp is not available after this call.

The closedir() function returns 0 on success.

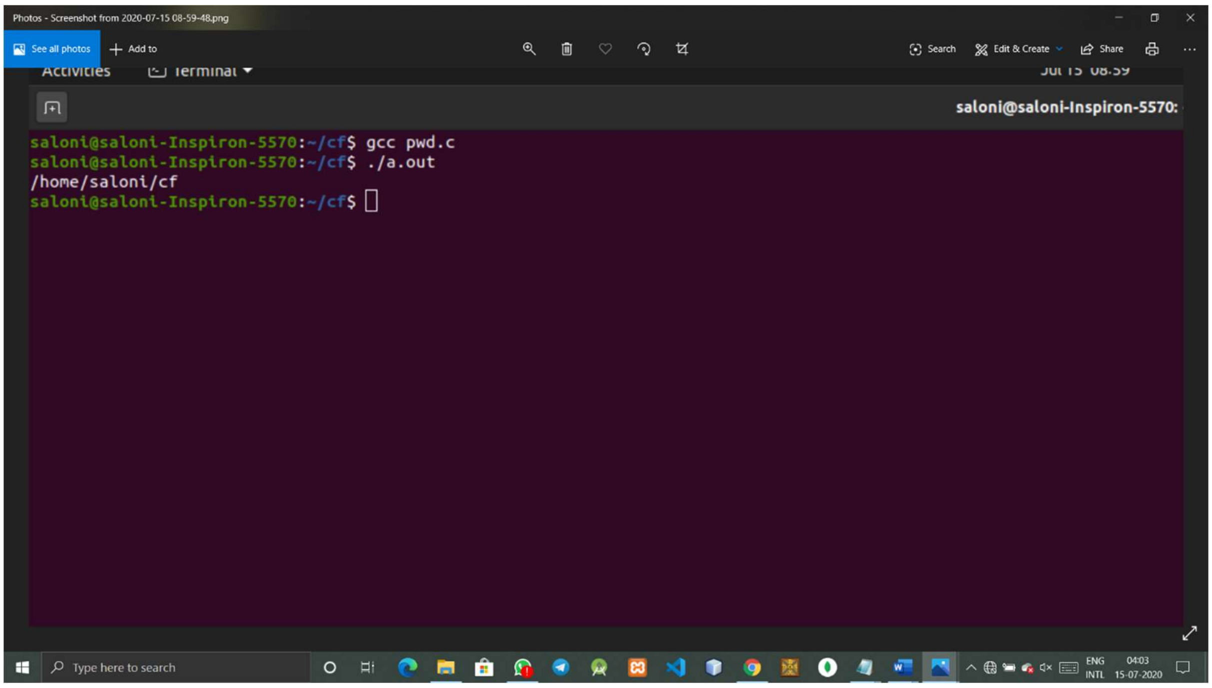
On error, -1 is returned, and errno is set appropriately.

**CODE:**

1. **Implement "pwd" command using system calls.**

```
#include<unistd.h>
#include<stdio.h>
int main()
{
        char path[100],*x;
        x=getcwd(path,sizeof(path));
        printf("%s\n",path);
        return 0;
}
```

**OUTPUT:**



```
saloni@saloni-Inspiron-5570:~/cf$ gcc pwd.c
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
/home/saloni/cf
saloni@saloni-Inspiron-5570:~/cf$ 
```

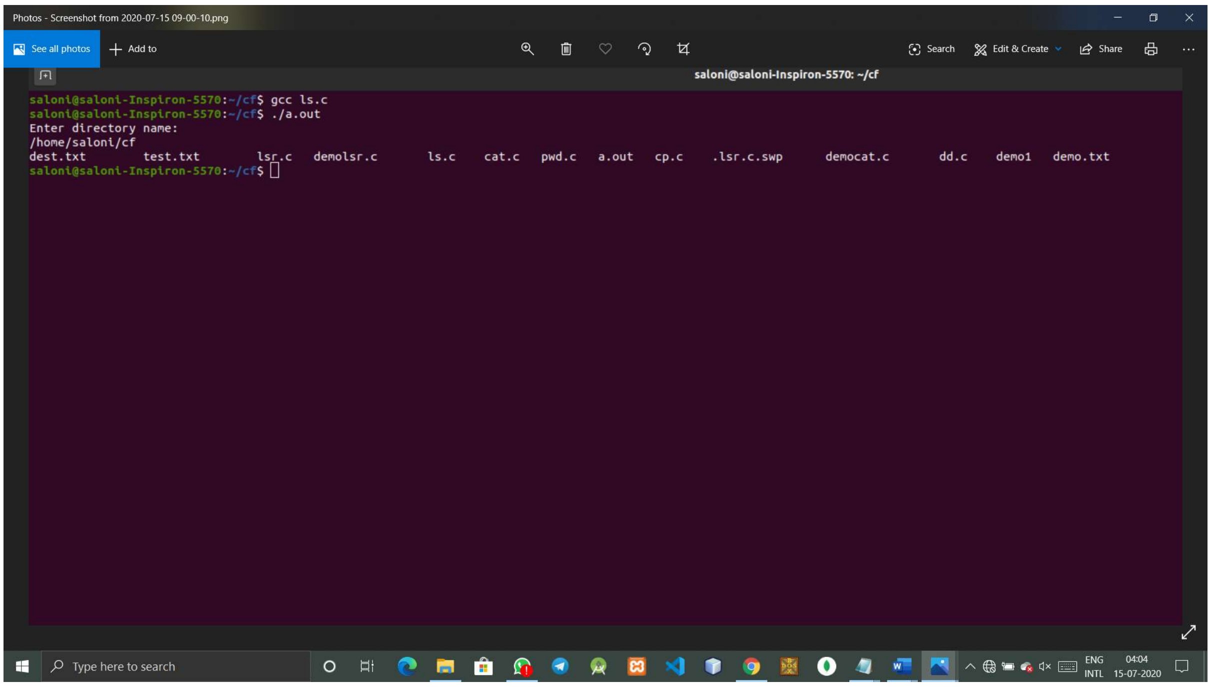
**DESC:**

This will give our present workind directory on which we are currently working on.

2. **Implement “ls” command.**

```
#include<stdio.h>
#include<dirent.h>
#include<stdlib.h>
#include<string.h>
int main(int argc,char *argv[])
{
        char arr[256];
        DIR *dp;
        struct dirent *dptr;
        printf("Enter directory name:\n");
        scanf("%s",arr);
        if((dp=opendir(arr))==NULL)
        {
                printf("EROOR");
                exit(1);
        }
        while(dptr=readdir(dp))
        {
                if(strcmp(dptr->d_name,".")==0){}
                else if(strcmp(dptr->d_name,"..")==0){}
                else
                {
                        printf("%s\t",dptr->d_name);
                }

        }
        printf("\n");

        closedir(dp);
}
```

**OUTPUT:**



```
saloni@saloni-Inspiron-5570:~/cf$ gcc ls.c
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
Enter directory name:
/home/saloni/cf
dest.txt        test.txt        lsr.c   demolsr.c       ls.c    cat.c   pwd.c   a.out   cp.c    .lsr.c.swp      democat.c       dd.c    demo1   demo.txt
saloni@saloni-Inspiron-5570:~/cf$ 
```

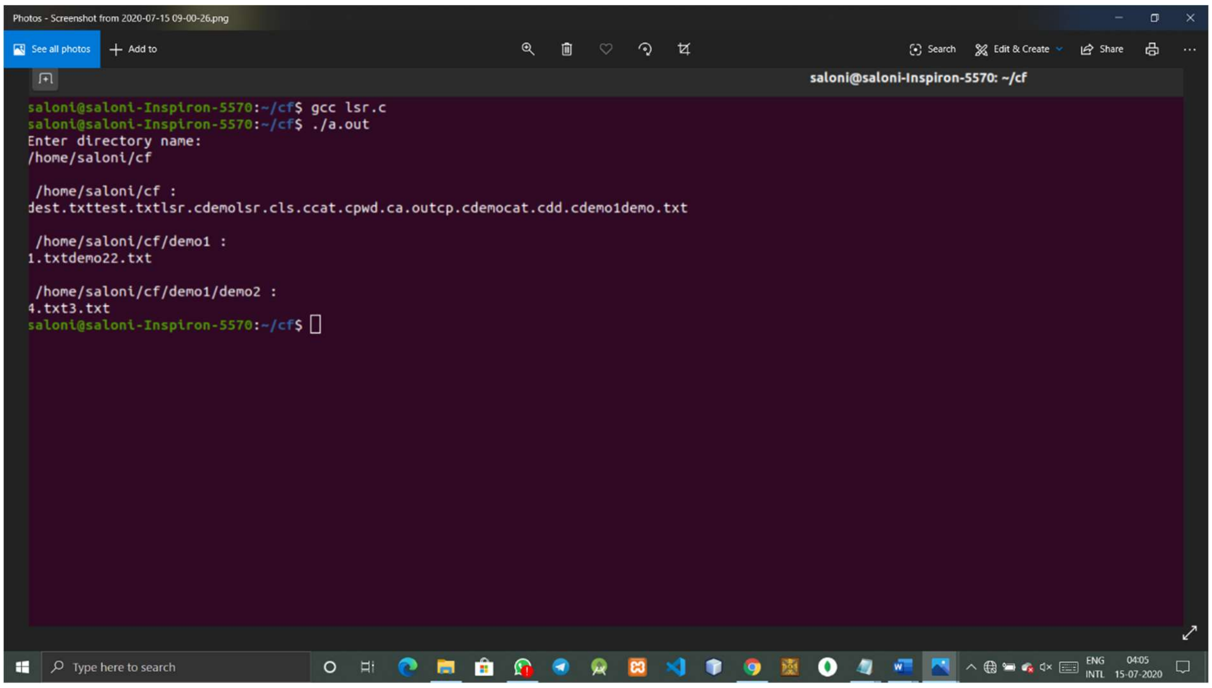
**DESC:**

Ls command will give all files and directory under our present worling directory except "." And ".."

3. **Implement "ls -R" using system calls.**

```
#include<stdio.h>
#include<dirent.h>
#include<stdlib.h>
#include<string.h>
void n(char []);
int main(int argc,char *argv[])
{
        char arr[256];
        DIR *dpd;
        struct dirent *dptrd;
        printf("Enter directory name:\n");
        scanf("%s",arr);
        n(arr);
}
void n(char *ar)
{
            DIR *dpd;
            char *dir[256];
            char *cur,temp[256];
            cur=ar;
            int i=0,j;
            struct dirent *dptrd;
            printf("\n %s :\n",cur);
        if((dpd=opendir(ar))==NULL)
       {
          printf("err");
       }
        while(dptrd=readdir(dpd))
       {

          if((dptrd->d_name[0])!='.')
             {

                 if(dptrd->d_type==4)
                 {
                         strcpy(temp,cur);
                         strcat(temp,"/");
                         strcat(temp,(dptrd->d_name));
                         dir[i]=malloc(strlen(temp)+1);
                         strcpy(dir[i],temp);
                         i++;

                 }
                 printf("%s",dptrd->d_name);

             }


       }
        closedir(dpd);
        printf("\n");
        for(j=0;j<i;j++)
       {
           n(dir[j]);
       }

 }
```

**OUTPUT:**



```
saloni@saloni-Inspiron-5570:~/cf$ gcc lsr.c
saloni@saloni-Inspiron-5570:~/cf$ ./a.out
Enter directory name:
/home/saloni/cf

 /home/saloni/cf :
dest.txttest.txtlsr.cdemolsr.cls.ccat.cpwd.ca.outcp.cdemocat.cdd.cdemo1demo.txt

 /home/saloni/cf/demo1 :
1.txtdemo22.txt

 /home/saloni/cf/demo1/demo2 :
4.txt3.txt
saloni@saloni-Inspiron-5570:~/cf$
```

**DESC:**

It will give all files and directory under pwd and if any directory found then it will recursively gives all the files and directory under that directory and so on.