Name: Abd**ullah Khan**

Roll no: 122A1002

Branch: CE

Batch: C1

Experiment #3

Aim: Implement basic commands of linux like ls, cp, mv and others using kernel APIs

Theory: The Linux kernel provides several interfaces to user-space applications that are used for different purposes and that have different properties by design. There are two types of application programming interface (API) in the Linux kernel that are not to be confused: the "kernel–user space" API and the "kernel internal" API.

The Linux API is the kernel–user space API, which allows programs in user space to access system resources and services of the Linux kernel.

a. **Implement "cp" using system calls.**

1) First check if both source & the destination files are received from the command line argument and exit if argc counter is not equal to 3.
There is also a check to handle "–help" option to print the usage of cpcmd.

2) Open the source file with read only flag set.

3) Open the destination file with the respective flags &
modes. O_WRONLY -> Open the file in write only mode
O_TRUNC -> Truncates the contents of the existing file
O_CREAT -> Creates a new file if it doesn't exist

S_IXUSR are file permissions for the current user ('X' can be R for read & W for write)
S_IXGRP are file permissions for the groups ('X' can be R for read & W for write)
S_IXOTH are file permissions for the groups ('X' can be R for read & W for write)

4) Start data transfer from source file to destination file till it reaches EOF (nbread == 0).

One of the most commonly used Linux commands is the ls command. The "ls" is a Linus shell command is which is used to print all the files and directories in the present directory in the form of a list.

b. **Implement "ls" command using system calls**

```
#include<stdio.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<dirent.h>
int main(int argc,char *argv[])
{
DIR *dp;
struct dirent *sd;
dp=opendir(argv[1]);
while((sd=readdir(dp))!=NULL)
{
printf("%s\t",sd->d_name);
}
closedir(dp);
}
```

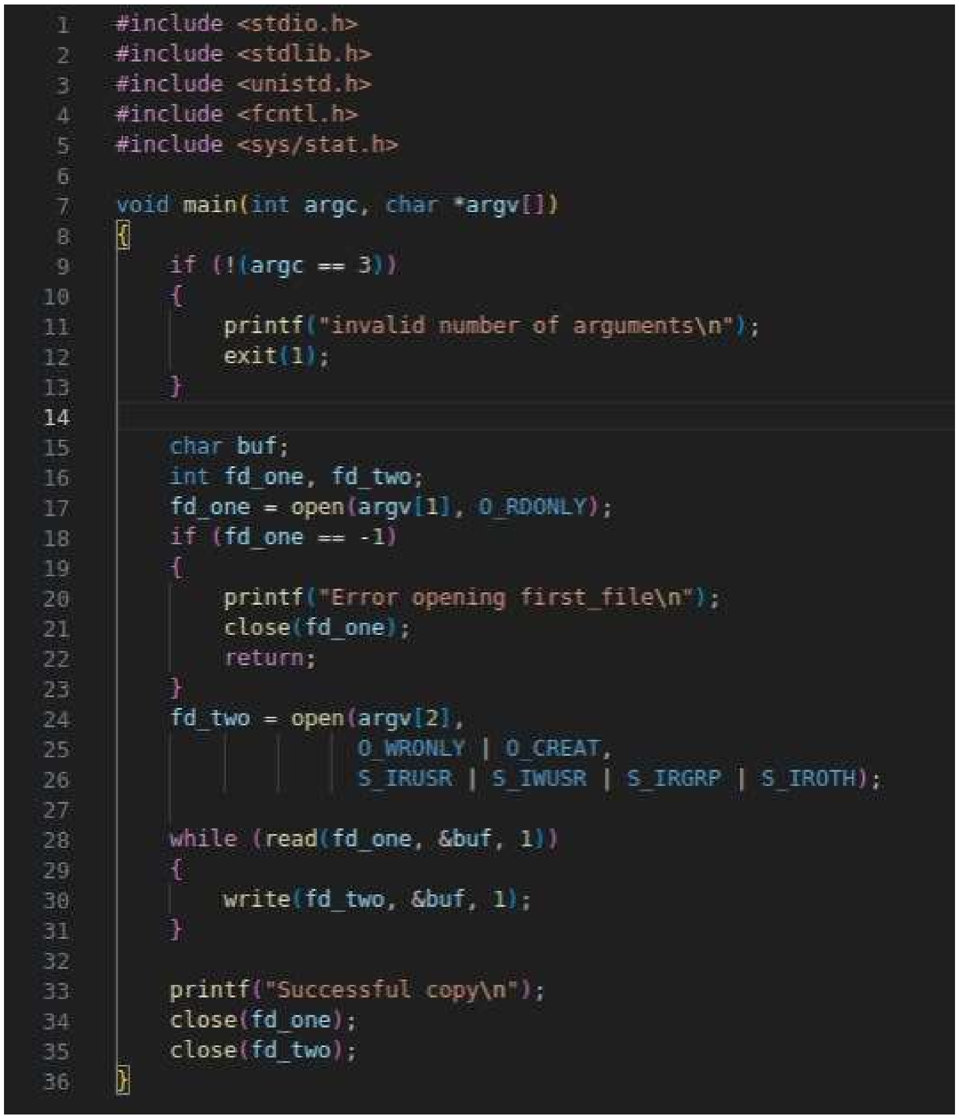
Output:

```
root@litmus:/home# gcc lscmd.c
root@litmus:/home#./a.out dirname
.        ..        f.txt     f1.c
```

c. Implement “mv” command using system calls

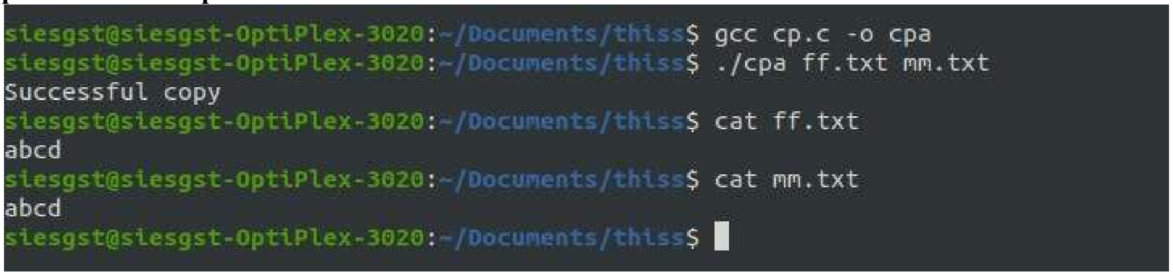
```
#include<stdio.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<dirent.h>
int main(int argc,char *argv[])
{
 int i, fd1,fd2;
 char *file1,*file2,buf[2];
 file1=argv[1];
 file2=argv[2];
 printf(“file1=%s file2=%s”, file1,file2);
 fd1=open(file1,O_RDONLY,0777);
 fd2=creat(file2,0777);
 while(i=read(fd1,buf,1)>0)
 write(fd2,buf,1);
 remove(file1);
 close(fd1);
 close(fd2);
 }
```

cp Command Code:



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

void main(int argc, char *argv[])
{
    if (!(argc == 3))
    {
        printf("invalid number of arguments\n");
        exit(1);
    }

    char buf;
    int fd_one, fd_two;
    fd_one = open(argv[1], O_RDONLY);
    if (fd_one == -1)
    {
        printf("Error opening first_file\n");
        close(fd_one);
        return;
    }
    fd_two = open(argv[2],
                  O_WRONLY | O_CREAT,
                  S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);

    while (read(fd_one, &buf, 1))
    {
        write(fd_two, &buf, 1);
    }

    printf("Successful copy\n");
    close(fd_one);
    close(fd_two);
}
```

cp Command Output:



```
siesgst@siesgst-OptiPlex-3020:~/Documents/thiss$ gcc cp.c -o cpa
siesgst@siesgst-OptiPlex-3020:~/Documents/thiss$ ./cpa ff.txt mm.txt
Successful copy
siesgst@siesgst-OptiPlex-3020:~/Documents/thiss$ cat ff.txt
abcd
siesgst@siesgst-OptiPlex-3020:~/Documents/thiss$ cat mm.txt
abcd
siesgst@siesgst-OptiPlex-3020:~/Documents/thiss$ 
```

Conclusion: Using the above sample programs, the basic commands can be executed successfully.