OS Lab 4:

Q.1. Implement UNIX commands “cat & mv” in C using System calls.
Ans:

1. **cat :** To implement cat, we first add a new file descriptor using open() system call, which return the value of new file descriptor entry created.
Using that value we start reading one-one character from the file using a character buffer and with the help of read() system call, and then write it to terminal(file descriptor 1) with the help of write() system call.
This is repeated for multiple input files using loop.

```
Applications ▾   Places ▾   Terminal ▾
root@kali:
File  Edit  View  Search  Terminal  Help
root@kali:~/Desktop/OS_L2# ls
alternate_characters  alternate_characters.c  cat_  cat_.c  mv_  mv_.c
root@kali:~/Desktop/OS_L2# ./cat_ cat_.c
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
#include<fcntl.h>

void main(int argc, char *argv[])
{
        // Proceed for one-one file
        for (int i = 1; i < argc; i++)
        {
                // Add new file descriptor
                int fd = open(argv[i], O_RDONLY);
                if (fd < 0)
                {
                        perror("Could not open file.");
                        return;
                }

                // Initialize buffer
                char *c = (char *) malloc(sizeof(1));

                // Read and write one-one character
                int a = read(fd, c, 1);
                while(a > 0)
                {
                        //printf("%c", *c);
                        write(1, c, 1);
                        a = read(fd, c, 1);
                }

                // Remove file descriptor
                close(fd);
        }
}
root@kali:~/Desktop/OS_L2#
```

2. **mv :** to implement mv, we repeat the same procedure as in cat; but here instead of writing to a terminal we are using a new file descriptor, which is returned by second open() call, which is used to open the file in which content is to moved.
If file is not already present or is already having content, that is handled by O_CREAT and O_TRUNC.
After moving, first file is deleted by using remove() system call.

```
Applications ▾   Places ▾   Terminal ▾
root@kali:
File  Edit  View  Search  Terminal  Help
root@kali:~/Desktop/OS_L2# ls
alternate_characters  alternate_characters.c  cat_  cat_.c  mv_  mv_.c
root@kali:~/Desktop/OS_L2# ./mv_ cat_.c temp.c
root@kali:~/Desktop/OS_L2# ls
alternate_characters  alternate_characters.c  cat_  mv_  mv_.c  temp.c
root@kali:~/Desktop/OS_L2# ./cat_ temp.c
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
#include<fcntl.h>

void main(int argc, char *argv[])
{
        // Proceed for one-one file
        for (int i = 1; i < argc; i++)
        {
                // Add new file descriptor
                int fd = open(argv[i], O_RDONLY);
                if (fd < 0)
                {
                        perror("Could not open file.");
                        return;
                }

                // Initialize buffer
                char *c = (char *) malloc(sizeof(1));

                // Read and write one-one character
                int a = read(fd, c, 1);
                while(a > 0)
                {
                        //printf("%c", *c);
                        write(1, c, 1);
                        a = read(fd, c, 1);
                }

                // Remove file descriptor
                close(fd);
        }
}
root@kali:~/Desktop/OS_L2# _
```

Q.2. Write a c program to read alternate character from the file and print it on terminal. Use open, read, write system calls.

Ans:

Alternate character can be reading first character then printing it, then reading one more character, but we won't print it. This second read character helps moving the pointer to the next character in file.

Then we repeat this process until file is empty, which we know when value returned by read() is 0.

This way alternate characters are printed starting from first character of file and skipping every second character.