**LAB 1 ASSIGNMENT**

**Name:** Birva Babaria

**Roll no.:** CE010

**ID:** 19CEUON064

**Aim:** Implementation of “cat” and “cp” command in C.

**Description of System calls:**

1. **Read system call**

**Library:** #include<unistd.h>

**Syntax:** ssize\_t read(int fd, void \*buff, size\_t count);

**Description:** read() attemps to read up to ‘count’ number of bytes from the file descriptor(fd) into the buffer starting at buff. If the execution succeeds, then the number of bytes is returned and the file position is advanced by the count of bytes. On error, -1 is returned and errorno is set appropriately.

**Example:** n = read(0, buff, sizeof(buff));

1. **Write system call**

**Library:** #include<unistd.h>

**Syntax:** ssize\_t write(int fd, const void \*buf, size\_t count);

**Description:** write() system call writes up to ‘count’ number of bytes pointed buff to the file referred to by the file descriptor(fd). If the execution succeeds, then the number of bytes written is returned. On error, -1 is returned and errorno is set appropriately.

**Example:** write(1, buff, n);

1. **Open system call**

**Library:** #include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

**Syntax:** int open(const char \*pathname, int flags);

OR

int open(const char \*pathname, int flags, mode\_t mode);

**Description:** open() system call opens the file from the pathname given in the argument and returns a small and non-negative integer for subsequent system calls. The argument ‘flags’ must include one of the access modes (O\_RDONLY (read only), O\_WRONLY (write only), O\_RDWR (read/write)).

In addition, zero or more file creation flags and file status flags can be bitwise or'd in flags.The file creation flags are O\_CLOEXEC, O\_CREAT, O\_DIRECTORY, O\_EXCL, O\_NOCTTY, O\_NOFOLLOW, O\_TRUNC, and O\_TTY\_INIT.

Argument mode specifies the permission to use in case new file is created. If the flag ‘O\_CREAT’ is not specified then mode is ignored.

**Example:** fd = open(“test.txt”, O\_RDONLY);

1. **Close system call**

**Library:** #include<unistd.h>

**Syntax:** int close(int fd);

**Description:** close() system call closes a file descriptor, so that it no longer points to any file and can be reused further. It returns 0 on success and on error -1 is returned and errorno is set appropriately.

**Example:** close(fd);

**1). Implement basic “cat” command using system calls.**

**CODE:**

//basic "cat" command using system calls.

#include<unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

int fd,n;

char buff[500];

//if file name is provided in command line argument then read that particular file

if(argc >= 2)

{

int i;

for(i=1;i<argc;i++)

{

fd = open(argv[i], O\_RDONLY);

n = read(fd, buff, sizeof(buff));

write(1, buff, n);

printf("\n--------------------\n");

close(fd);

}

}

//if filename is not present then program works as a 'cat' command (press ctrl+c to end)

else

{

while(1)

{

n = read(0, buff, sizeof(buff));

write(1, buff, n);

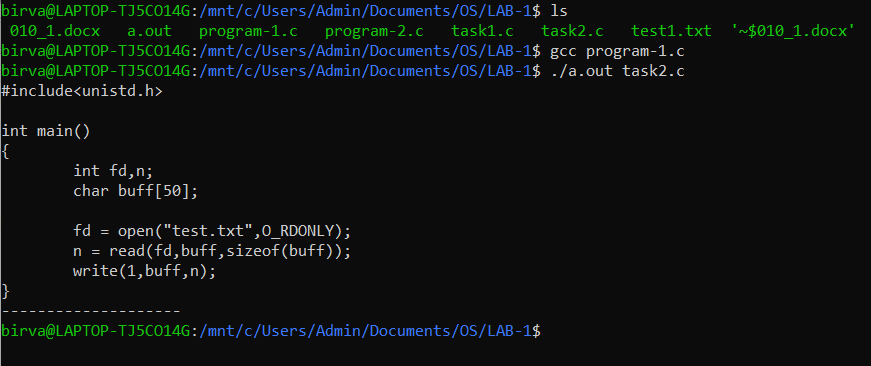
}

}

}

**OUTPUT:**

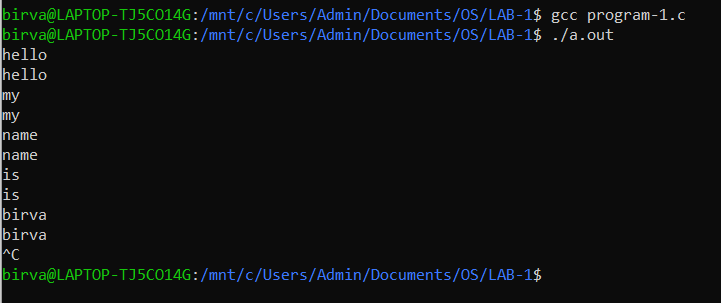
* One file name provided in command line argument.

****

* Multiple file names provided in command line argument.

****

* File name not provided in command line argument.



**2). Implement basic “cp” command using system calls.**

**CODE:**

//basic "cp" command using system calls.

#include<unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdio.h>

int main(int argc, char \*argv[])

{

int fd1,fd2,n;

char buff[500];

//if two file names are provided

if(argc == 3)

{

//open and read from file 1

fd1 = open(argv[1], O\_RDONLY);

n = read(fd1, buff, sizeof(buff));

//open and write in file 2

fd2 = open(argv[2], O\_WRONLY | O\_CREAT, 666);

write(fd2, buff, n);

close(fd1);

close(fd2);

}

else

{

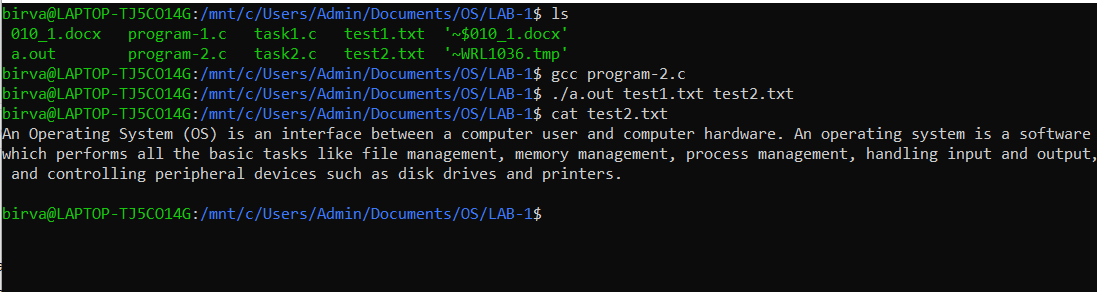
printf("INVALID NUMBER OF ARGUMENTS ENTERED!!\n");

}

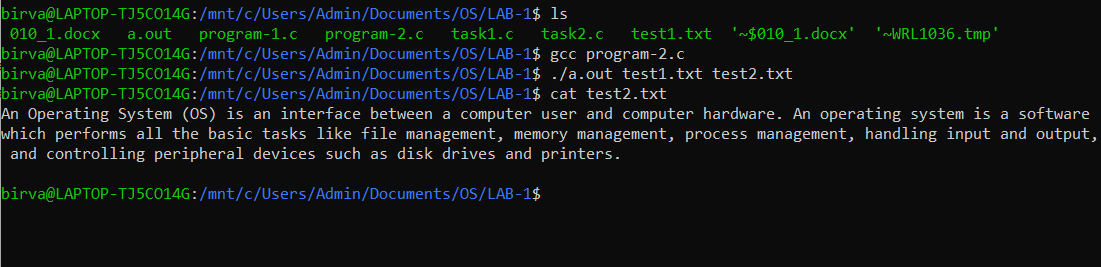
}

**OUTPUT:**

* File 2 in command line argument exist.

****

* File 2 in command line argument does not exist (created on its own).

****