

Laboratory 2

Title of the Laboratory Exercise: Programs using file management system calls

1. Introduction and Purpose of Experiment

A system call is a programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. There are different types of system calls developed for various purposes. They are mainly classified as process management, file management, directory management. By solving the problems students will be able to apply file management system calls

Aim and Objectives

Aim

- To develop programs involving file management system calls

Objectives

At the end of this lab, the student will be able to

- Use different file management system calls
- Apply different system calls wherever required
- Create C programs using file management system calls

2. Experimental Procedure

- i. Analyse the problem statement
- ii. Design an algorithm for the given problem statement and develop a flowchart/pseudo-code
- iii. Implement the algorithm in C language
- iv. Compile the C program
- v. Test the implemented program
- vi. Document the Results
- vii. Analyse and discuss the outcomes of your experiment

3. Questions

Implement the following command in C

Implement copy command (cp) to copy a file content to other file using file management system calls

4. Calculations/Computations/Algorithms:-

STEP 1: Start

STEP 2: buff \leftarrow string of size 100

STEP 3: inFile \leftarrow in_file.txt file descriptor

STEP 4: outFile \leftarrow out_file.txt file descriptor

STEP 5: bytesRead \leftarrow 0, bytesWritten \leftarrow 0

STEP 6: while bytesRead = read(inFile) and not EOF do

6.1: bytesWritten \leftarrow write to outFile

STEP 7: if bytesWritten is greater than 0, then

7.1: display success message

STEP 8: Stop

5. Presentation of Results:-

Program:-

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

void main()
{
    char buff[100];

    int inFile = open("file1.txt", O_RDONLY);
    int outFile = open("file2.txt", O_WRONLY);

    int bytesRead, bytesWritten;

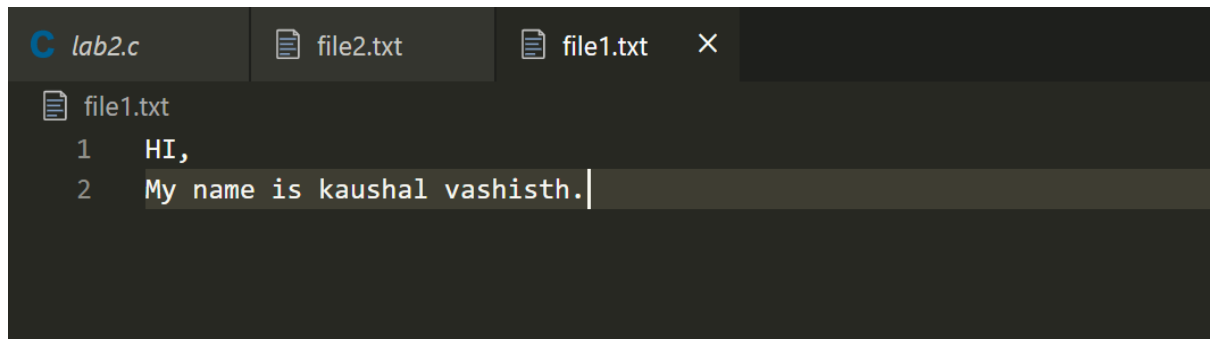
    while ((bytesRead = read(inFile, buff, 100)) != 0)
        bytesWritten = write(outFile, buff, bytesRead);

    if (bytesWritten >= 0)
        printf("Contents copied successfully.\n");
}
```

Output:-

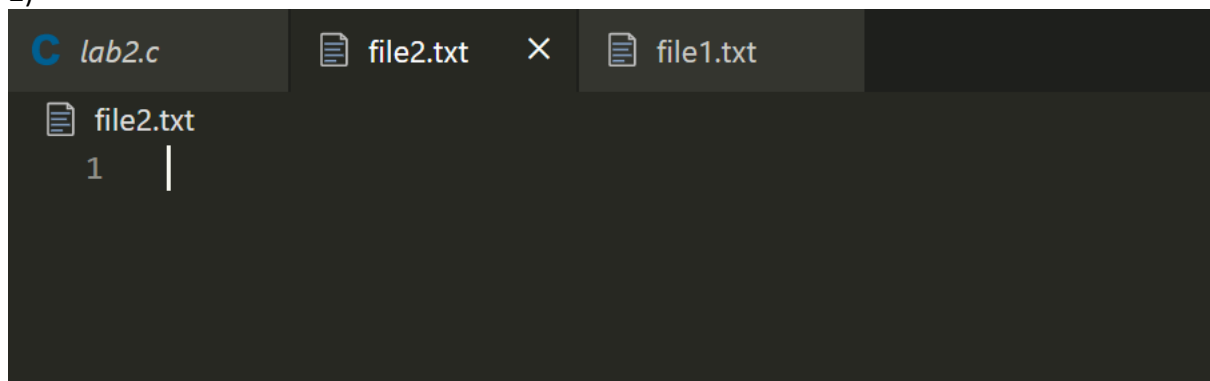
Before Executing the Program:-

1)



A screenshot of a code editor with three tabs: lab2.c, file2.txt, and file1.txt. The file1.txt tab is active, showing two lines of text: "1 HI," and "2 My name is kaushal vashisth." The cursor is at the end of the second line.

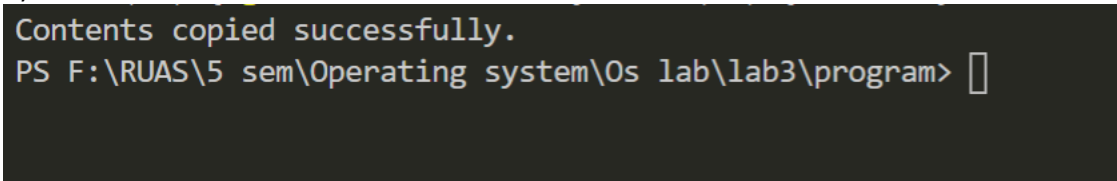
2)



A screenshot of a code editor with three tabs: lab2.c, file2.txt, and file1.txt. The file2.txt tab is active, showing one line of text: "1 |". The cursor is at the end of the line.

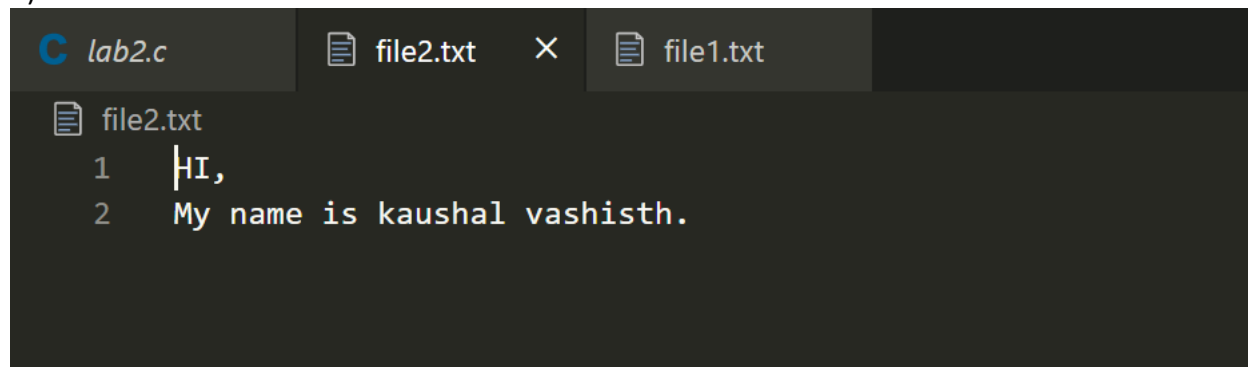
After executing the program:-

1)



A screenshot of a command prompt window showing the message "Contents copied successfully." followed by the command prompt "PS F:\RUAS\5 sem\Operating system\Os lab\lab3\program>".

2)



A screenshot of a code editor with three tabs: lab2.c, file2.txt, and file1.txt. The file2.txt tab is active, showing two lines of text: "1 HI," and "2 My name is kaushal vashisth." The cursor is at the end of the second line.

6. Analysis and Discussions:-

The cp command can be used to copy files or it's contents into another file anywhere in the directory.

7. Conclusions

Different test cases show how the program reacts when subjected to copy their contents.

8. Comments

1. Limitations of Experiments

Even when the file is not created and if command is not used, still the build is successful due to creation of binary file which the user can't see. This may prove to be a confusion

2. Recommendations :-Can use exception handling.