Name: Subhendu Maji Roll number: 18ETCS002121

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 ← string of size 100

STEP 3: inFile ← in_file.txt file descriptor

STEP 4: outFile ← out_file.txt file descriptor

STEP 5: bytesRead ← 0, bytesWritten ← 0

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

      6.1: bytesWritten ← write to outFile

STEP 7: if bytesWritten is greater than 0, then

      7.1: display success message

STEP 8: Stop
```

5. Presentation of Results

lab2.c > ...

```c
#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");
}
```

Figure 4 Source Code

```
makefile
1  lab:
2      @gcc lab2.c -lm
3      @./a.out
4
5  clean:
6      @rm *.out
7
8  .PHONY: clean
9
10
```

Figure 5 Makefile

```
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2  cat file1.txt
Hi, My name is Subhendu Maji. I make awesome things for awesome people.
I am a software engineer with 1 year of experience residing in Delhi, India.%
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2  cat file2.txt
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2
```

Figure 6 Files before Execution

```
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2  make lab
Contents copied successfully.
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2  cat file1.txt
Hi, My name is Subhendu Maji. I make awesome things for awesome people.
I am a software engineer with 1 year of experience residing in Delhi, India.%
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2  cat file2.txt
Hi, My name is Subhendu Maji. I make awesome things for awesome people.
I am a software engineer with 1 year of experience residing in Delhi, India.%
 subhendu@LAPTOP-AL8CTHTV  /mnt/d/RUAS-sem-05/OS/lab2
```

Figure 7 Execution and file copied successfully.

6. Analysis and Discussions

In this experiment, System calls are used to copy the contents from one file to another file. To open a file, the `open()` system call was used. The `open()` system call takes the file path and the file mode as arguments. The function returns a file descriptor.

The `read()` system call is used to read a file that is open in read mode. The `read()` call takes the file descriptor to be read from, the buffer to store the data that is read and the amount of bytes to read. The function returns the number of bytes that was successfully read.

The `write()` system call is used to write data to a file that is open in write mode. The `write()` system call takes the output file descriptor, buffer containing the data and the number of bytes to write. The function returns the number of bytes that was successfully written to the file.

7. Conclusions

In this experiment, system calls were used to open, read and write to a file. These calls were used to make a mock 'cp' command. The program was implemented in C.

8. Comments

1. Limitations of Experiments

The `read()` system call reads from the file sequentially.

2. Limitations of Results

The use of system calls to implement the 'cp' command was learned.

3. Recommendations

`readv()` and `writev()` can be used, which read and write from multiple buffers at once.