

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

copy_file(source, destination)

1. open(source, READ_ONLY)
2. open(destination, WRITE_ONLY | CREATE)
3. if either of it fails, return FAILURE
4. create an empty buffer array
5. read 9216 bytes of source and write strlen(source) to destination
6. repeat until EOF is reached
7. close source and destination file_descriptors

5. Presentation of Results

Test Case

To Show the error Message when Copying from Source file to destination File if Source will is not found

Input

```
1 //Program Done By Deepk R 18ETCS002041
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <fcntl.h>
5 #include <unistd.h>
6 #define buf_len 9216
7 int main() {
8     int fd1,fd2,in_fd,out_fd,cd1,cd2;
9     char *buf[buf_len];
10    fd1 = open("C:/Users/HP/Desktop/File1.txt", O_RDONLY);
11    if(fd1>0)
12        printf("\nFile1.txt file located!!");
13    else
14        printf("\nFile1.txt file missing!!");
15
16    fd2 = open("C:/Users/HP/Desktop/File2.txt",O_WRONLY | O_CREAT);
17    if(fd2>0)
18        printf("\nFile2.txt file created!!");
19    else
20        printf("\nFile2.txt file missing!!");
21
22    while(in_fd = read(fd1,buf,buf_len) > 0) {
23        out_fd = write(fd2,buf,buf_len);
24        if(out_fd!=in_fd)
25            break;
26    }
27    if(in_fd==-1)
28        printf("\nCannot read from File1.txt as file is missing!!");
29    if(out_fd==-1)
30        printf("\nCannot write to File2.txt as file is missing!!");
31    /* Closing all files*/
32    cd1 = close(fd1);
33    cd2 = close(fd2);
34    if(cd1==-1 || cd2==-1) {
35        printf("\nData from File1.txt was not copied to File2.txt\n");
36    }
37    else
38        printf("\nData from File1.txt was copied to File2.txt\n");
39    return 0;
40 }
```

Output

Output x



lab_2 (Build, Run) x

lab_2 (Run) x



File1.txt file missing!!

File2.txt file created!!

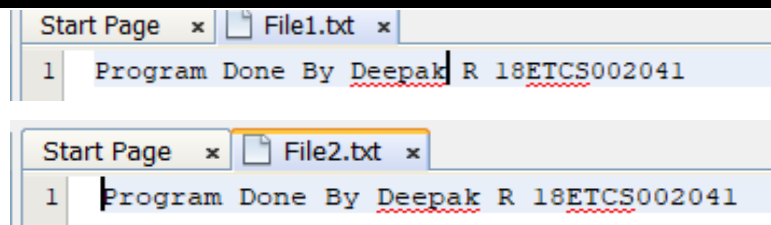
Data from File1.txt was not copied to File2.txt

Test Case

To Copy the Content of the one file to another file when both Files are already Created in Same Folder

Input

```
//Program By Deepak R
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
FILE *fd1,*fd2;
char buf[100];
int main(int argc, char** argv) {
    fd1 = open("File1.txt",O_RDONLY);
    fd2 = open("File2.txt",O_WRONLY);
    read(fd1,buf,sizeof(buf));
    write(fd2,buf,sizeof(buf));
    close(fd1);
    close(fd2);
    return (EXIT_SUCCESS);
}
```

Output

Start Page x File1.txt x

1 Program Done By Deepak R 18ETCS002041

Start Page x File2.txt x

1 Program Done By Deepak R 18ETCS002041

Test Case

To Copy the Content of the one file to another file Even if Destination file Doesn't Exist It gets created

Because of O_CREAT function

Input

```
7 //Program By Deepak R
8 #include <stdio.h>
9 #include <stdlib.h>
10 #include <fcntl.h>
11 FILE *fd1,*fd2;
12 char buf[100];
13 int main(int argc, char** argv) {
14     fd1 = open("File1.txt",O_RDONLY);
15     fd2 = open("File2.txt",O_WRONLY | O_CREAT );
16     read(fd1,buf,sizeof(buf));
17     write(fd2,buf,sizeof(buf));
18     close(fd1);
19     close(fd2);
20     return (EXIT_SUCCESS);
21 }
```

Output

Start Page x File1.txt x
1 Program Done By Deepak R 18ETCS002041

Start Page x File2.txt x
1 Program Done By Deepak R 18ETCS002041

Test Case

To Copy the Content of the one file to another file By giving the path Even if Destination file Doesn't Exist It gets created Because of O_CREAT function

Input

```
Start Page x File1.txt x main.c x
1  /*
2   * File:   main.c
3   * Author: HP
4   *
5   * Created on 27 October, 2020, 2:49 PM
6   */
7  //Program By Deepak R
8  #include <stdio.h>
9  #include <stdlib.h>
10 #include <fcntl.h>
11 FILE *fd1,*fd2;
12 char buf[100];
13 int main(int argc, char** argv) {
14     fd1 = open("File1.txt",O_RDONLY);
15     fd2 = open("C:/Users/HP/Desktop/File2D.txt",O_WRONLY | O_CREAT );
16     read(fd1,buf,sizeof(buf));
17     write(fd2,buf,sizeof(buf));
18     close(fd1);
19     close(fd2);
20     return (EXIT_SUCCESS);
21 }
```

Output

```
Start Page x File1.txt x
1  Program Done By Deepak R 18ETCS002041
```



File2D - Notepad

File Edit Format View Help

Program Done By Deepak R 18ETCS002041

Test CaseTo Copy the Content of the one file to another file from the file with Multiple lines**Input**

```
1  #Program By Deepak R !8ETCS002041
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <fcntl.h>
5  #include <unistd.h>
6  #define buf_len 9216
7
8  int main() {
9      int fd1,fd2,in_fd,out_fd,cd1,cd2;
10     char *buf[buf_len];
11     fd1 = open("C:/Users/HP/Desktop/File1.txt", O_RDONLY);
12     if(fd1>0)
13         printf("\nFile1.txt file located!!");
14     else
15         printf("\nFile2.txt file missing!!");
16
17     fd2 = open("C:/Users/HP/Desktop/File2.txt",O_WRONLY | O_CREAT);
18     if(fd2>0)
19         printf("\nFile2.txt file created!!");
20     else
21         printf("\nFile2.txt file missing!!");
22
23     while(in_fd = read(fd1,buf,buf_len) > 0) {
24         out_fd = write(fd2,buf,buf_len);
25         if(out_fd!=in_fd)
26             break;
27     }
28     if(in_fd==-1)
29         printf("\nCannot read from File1.txt as file is missing!!");
30     if(out_fd==-1)
31         printf("\nCannot write to File2.txt as file is missing!!");
32     /* Closing all files*/
33     cd1 = close(fd1);
34     cd2 = close(fd2);
35     if(cd1==-1 || cd2==-1) {
36         printf("\nData from File1.txt was not copied to File2.txt\n");
37     }
38     else
39         printf("\nData from File1.txt was copied to File2.txt\n");
40     return 0;
41 }
```

Output

Output x

lab_2 (Build, Run) x lab_2 (Run) x

File1.txt file located!!
File2.txt file created!!
Data from File1.txt was copied to File2.txt

File1 - Notepad

File Edit Format View Help

Deepak R
!8ETCS002041
Ramaiah University of Applied Sciences

File2 - Notepad

File Edit Format View Help

Deepak R
!8ETCS002041
Ramaiah University of Applied Sciences

Test Case

To Show the error Message when Copying from Source file to destination File if Source and destination file will is not found

Input

```
Start Page x File1.txt x main.c x main.c x
//Program Done By Deepak R 18ETCS002041
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <fcntl.h>
5 #include <unistd.h>
6 #define buf_len 9216
7 int main() {
8     int fd1,fd2,in_fd,out_fd,cd1,cd2;
9     char *buf[buf_len];
10    fd1 = open("C:/Users/HP/Desktop/File1.txt", O_RDONLY);
11    if(fd1>0)
12        printf("\nFile1.txt file located!!");
13    else
14        printf("\nFile1.txt file missing!!");
15
16    fd2 = open("C:/Users/HP/Desktop/File2.txt",O_WRONLY );
17    if(fd2>0)
18        printf("\nFile2.txt file created!!");
19    else
20        printf("\nFile2.txt file missing!!");
21
22    while(in_fd = read(fd1,buf,buf_len) > 0) {
23        out_fd = write(fd2,buf,buf_len);
24        if(out_fd!=in_fd)
25            break;
26    }
27    if(in_fd===-1)
28        printf("\nCannot read from File1.txt as file is missing!!");
29    if(out_fd===-1)
30        printf("\nCannot write to File2.txt as file is missing!!");
31    /* Closing all files*/
32    cd1 = close(fd1);
33    cd2 = close(fd2);
34    if(cd1===-1 || cd2===-1) {
35        printf("\nData from File1.txt was not copied to File2.txt\n");
36    }
37    else
38        printf("\nData from File1.txt was copied to File2.txt\n");
39    return 0;
40 }
```

Output

```
Output x
lab_2 (Build, Run) x lab_2 (Run) x
File1.txt file missing!!
File2.txt file missing!!
Data from File1.txt was not copied to File2.txt
```


Explanation:

Here the content of file1.txt is copied to file1.txt, initially the file2.txt file does not exist, hence the function defined in the program creates the files, reads the file file1.txt and writes the content until EOF bit is reached. If the program encounters any error during opening of either the source or destination file, it displays the error and closes the source/destination file as necessary.

If the destination file exists and already has some content, then it is overwritten with the source file content.

6. Analysis and Discussions

open, openat, creat - open and possibly create a file

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

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

int creat(const char *pathname, mode_t mode);

int openat(int dirfd, const char *pathname, int flags);
int openat(int dirfd, const char *pathname, int flags, mode_t mode);
```

Description:

The **open()** system call opens the file specified by *pathname*. If the specified file does not exist, it may optionally (if **O_CREAT** is specified in *flags*) be created by **open()**.

A call to **open()** creates a new *open file description*, an entry in the system-wide table of open files. The open file description records the file offset and the file status flags. A file descriptor is a reference to an open file description; this reference is unaffected if *pathname* is subsequently removed or modified to refer to a different file.

The argument *flags* must include one of the following *access modes*: **O_RDONLY**, **O_WRONLY**, or **O_RDWR**. These requests opening the file read-only, write-only, or read/write, respectively.

7. Conclusions

The *open()* function shall establish the connection between a file and a file descriptor. It shall create an open file description that refers to a file and a file descriptor that refers to that open file description. The file descriptor is used by other I/O functions to refer to that file. The *path* argument points to a pathname naming the file.

Upon successful completion, the function shall open the file and return a non-negative integer representing the lowest numbered unused file descriptor. Otherwise, -1 shall be returned and *errno* set to indicate the error. No files shall be created or modified if the function returns -1.

Some errors that may occur during the program execution are:

1. Search permission is denied on a component of the path prefix.
2. The file already exists and the permissions specified by *oflag* is denied.
3. The length of the path argument exceeds *PATH_MAX*, or the pathname is longer than *NAME_MAX*.
4. The named file resides on a *READ_ONLY* file system.
5. The named file is a character special file or a block special file.

8. Comments

1. Limitations of Experiments

The experiment is limited to mimicking the actions of *cp* command with only using *open* and *close* syscalls, this raises issues with some of the errors that may occur during the execution that needs to be taken care of separately.

2. Limitations of Results

The implemented program assumes that the file does not contain null characters in the middle of the file, here the program will fail to copy the contents of the file properly.

3. Learning happened

The concept of syscalls for reading and writing files in Linux is learnt.

4. Recommendations

To further enhance the program, user permissions from the source files should be copied to the destination file.

File permissions of the source file needs to be checked properly and appropriate error messages should be shown to the user in case of errors.