

Name: K.Saikrishna

Reg-No: 192311106

2. Identify the system calls to copy the content of one file to another and illustrate the same using a C program.

Aim

To understand file handling using system calls in C by copying the content of one file to another.

Algorithm

1. Open the source file in read-only mode using the `open()` system call.
2. Open (or create) the destination file in write mode using `open()`.
3. Use a loop to read the content of the source file in chunks using the `read()` system call.
4. Write the read content into the destination file using the `write()` system call.
5. Continue until the end of the source file is reached.
6. Close both files using the `close()` system call.

Procedure

1. Use `open()` to handle file descriptors for the source and destination files.
2. Check for errors (e.g., if the files cannot be opened).
3. Use a buffer to read data from the source file and write it to the destination file.
4. Handle edge cases like empty files or read/write errors.
5. Ensure both files are properly closed at the end of the operation.

Code:

```
#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#define BUFFER_SIZE 1024

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

    int source, destination;

    char buffer[BUFFER_SIZE];
```

```
    ssize_t bytesRead, bytesWritten;

    if (argc != 3) {

        write(STDERR_FILENO, "Usage: ./copyfile <source> <destination>\n", 41);

        exit(1);

    }

    source = open(argv[1], O_RDONLY);

    if (source < 0) {

        perror("Error opening source file");

        exit(1);

    }

    destination = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0644);

    if (destination < 0) {

        perror("Error opening destination file");

        close(source);

        exit(1);

    }

    while ((bytesRead = read(source, buffer, BUFFER_SIZE)) > 0) {

        bytesWritten = write(destination, buffer, bytesRead);

        if (bytesWritten != bytesRead) {

            perror("Error writing to destination file");

            close(source);

        }

        close(destination);

        exit(1);

    }

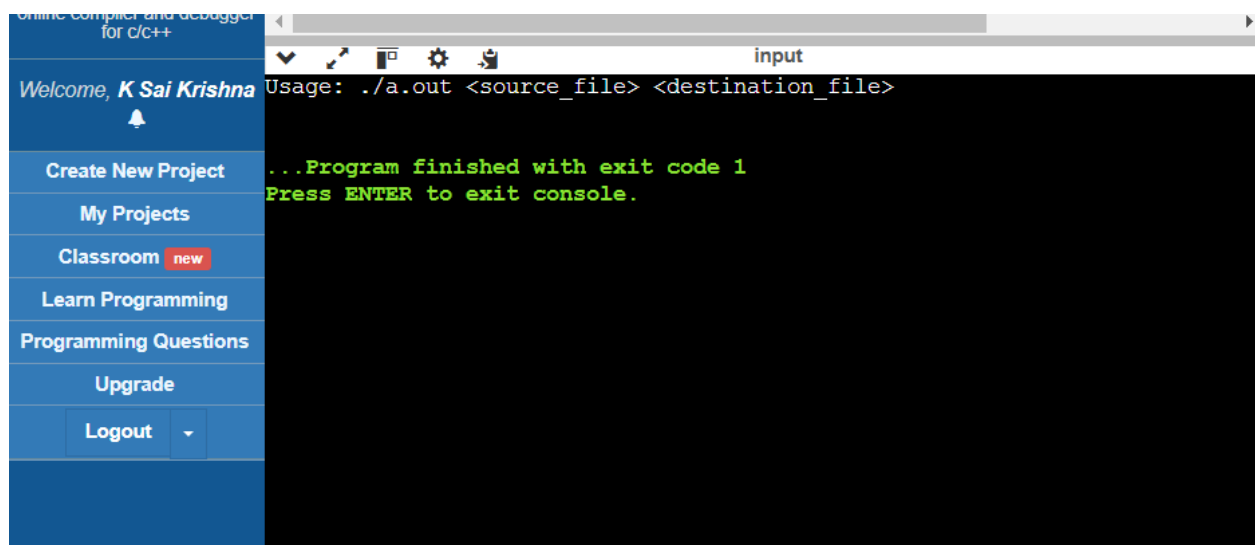
}
```

```
    }  
}  
  
if (bytesRead < 0)  
    perror("Error reading source file");  
  
close(source);  
  
close(destination);  
  
return 0;  
}
```

Result

The program successfully copies the content of the source file into the destination file using system calls, demonstrating efficient file handling in C.

Output:



The screenshot shows an online compiler interface for C/C++. On the left is a sidebar with navigation links: "Welcome, K Sai Krishna", "Create New Project", "My Projects", "Classroom new", "Learn Programming", "Programming Questions", "Upgrade", and "Logout". The main area displays the program's output in a terminal window. The output text is: "Usage: ./a.out <source_file> <destination_file>", "...Program finished with exit code 1", and "Press ENTER to exit console." The terminal window has a title bar that says "input".

```
Usage: ./a.out <source_file> <destination_file>  
  
...Program finished with exit code 1  
Press ENTER to exit console.
```