

Name: T. Nikhil Kumar Reddy

Reg-No: 192372024

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:

```

main.c
1  #include <fcntl.h>      T.NIKHIL
2  #include <unistd.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  #define BUFFER_SIZE 1024
7
8  int main(int argc, char *argv[]) {
9      int source, destination;
10     char buffer[BUFFER_SIZE];
11     ssize_t bytesRead, bytesWritten;
12
13     if (argc != 3) {
14         write(STDERR_FILENO, "Usage: ./copyfile <source> <destination>\n", 41);
15         exit(1);
16     }
17
18     source = open(argv[1], O_RDONLY);
19     if (source < 0) {
20         perror("Error opening source file");
21         exit(1);
22     }
23
24     destination = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0644);
25     if (destination < 0) {
26         perror("Error opening destination file");
27         close(source);
28         exit(1);
29     }
30
31     while ((bytesRead = read(source, buffer, BUFFER_SIZE)) > 0) {
32         bytesWritten = write(destination, buffer, bytesRead);
33         if (bytesWritten != bytesRead) {
34             perror("Error writing to destination file");
35             close(source);
36             close(destination);
37             exit(1);
38         }
39     }
40
41     if (bytesRead < 0)
42         perror("Error reading source file");
43
44     close(source);
45     close(destination);
46
47     return 0;
48 }
49
Usage: ./copyfile <source> <destination>

...Program finished with exit code 1
Press ENTER to exit console.

```