main.c

Share Run

```c
#include <stdio.h>
#include <unistd.h>
int main()
{
    int pid = fork();
    if (pid > 0)
        printf("Parent process: PID = %d\n", getpid());
    else if (pid == 0)
        printf("Child process: PID = %d\n", getpid());
    else
        printf("Fork failed!\n");
return 0;
}
```

Output

```
Parent process: PID = 6814
Child process: PID = 6815


=== Code Execution Successful ===
```

main.c

Share Run

Output

```c
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#define BUFFER_SIZE 1024
int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <source_file> <destination_file>\n", argv[0]);
        exit(1);
    }
    int src_fd = open(argv[1], O_RDONLY);
    if (src_fd < 0) {
        perror("Error opening source file");
        exit(1);
    }
    int dest_fd = open(argv[2], O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (dest_fd < 0) {
        perror("Error opening/creating destination file");
        close(src_fd);
        exit(1);
    }
    char buffer[BUFFER_SIZE];
    ssize_t bytes_read, bytes_written;
    while ((bytes_read = read(src_fd, buffer, BUFFER_SIZE)) > 0) {
        bytes_written = write(dest_fd, buffer, bytes_read);
        if (bytes_written != bytes_read) {
            perror("Error writing to destination file");
            close(src_fd);
            close(dest_fd);
            exit(1);
        }
    }
    if (bytes_read < 0) {
        perror("Error reading source file");
    }
    close(src_fd);
    close(dest_fd);
    printf("File copied successfully.\n");
    return 0;
}
```

```
Usage: /tmp/hf8wJDPmO9/main.o <source_file> <destination_file>


=== Code Exited With Errors ===
```