Client-Server Program

Client :

```
#include<stdio.h>       // Include standard input/output library
#include<stdlib.h>      // Include standard library for general functions
#include<sys/types.h>   // Include definitions for data types used in system calls
#include<sys/stat.h>    // Include definitions for file status
#include<unistd.h>      // Include standard symbolic constants and types
#include<fcntl.h>       // Include file control options
#include<string.h>      // Include string handling functions

int main() {
   puts("\n\tClient - Listening\n"); // Print a message indicating the client is listening

   // Create two named FIFOs (first-in-first-out special files) for communication
   int code6 = mkfifo("fifo6.txt", 0666); // FIFO for reading
   int code7 = mkfifo("fifo7.txt", 0666); // FIFO for writing

   char strMessage[5000]; // Buffer for messages

   // Check if FIFO creation was successful
   if(code6 == -1)
      perror("\n\tmkfifo6 returned an error - file may already exist\n"); // Print error if FIFO6 failed
   if(code7 == -1)
      perror("\n\tmkfifo7 returned an error - file may already exist\n"); // Print error if FIFO7 failed

   // Open the FIFOs for reading and writing
   int fd = open("fifo6.txt", O_RDONLY); // Open FIFO6 for reading
   int fd2 = open("fifo7.txt", O_WRONLY); // Open FIFO7 for writing

   // Check if the FIFO for reading was opened successfully
   if(fd == -1) {
      perror("Cannot open FIFO6 for read"); // Print error message
      return EXIT_FAILURE; // Exit with failure status
   }

   // Check if the FIFO for writing was opened successfully
   if(fd2 == -1) {
      perror("Cannot open FIFO7 for write"); // Print error message
      return EXIT_FAILURE; // Exit with failure status
   }

   puts("FIFO OPEN"); // Indicate that FIFOs are open

   // Buffer to read the incoming message
   char stringBuffer[5000];
   memset(stringBuffer, 0, 5000); // Initialize buffer to zero

   int res; // Variable for read results
   char Len; // Variable to hold the length of the message
```

```
    // Main loop for reading and processing messages
    {
        res = read(fd, &Len, 1); // Read the length of the message (1 byte)

        // Read the actual message into the buffer
        read(fd, stringBuffer, Len); // Read string characters

        stringBuffer[(int)Len] = 0; // Null-terminate the string
        printf("\nClient Received: %s\n", stringBuffer); // Print the received message

        int j = 0, w = 0, line = 0; // Counters for words, characters, and lines

        // Count words, characters, and lines in the received message
        while(stringBuffer[j] != '\0') {
            char ch = stringBuffer[j];
            if((ch == ' ') || (ch == '\n')) { // Check for spaces and newlines
                w++; // Increment word count
                if(ch == '\n') // If newline is found, increment line count
                    line++;
            }
            j++; // Move to the next character
        }

        // Prepare strings for output
        char LC = (char)strlen(strMessage); // Get length of the message
        char str1[256], str2[256], str3[256]; // Buffers for formatted output

        sprintf(str1, " No.of Words : %d:::", w); strcat(strMessage, str1); // Append word count to
message
        sprintf(str2, " No.of Characters: %d:::", (j - 1)); strcat(strMessage, str2); // Append character
count
        sprintf(str3, " No.of Lines: %d", line); strcat(strMessage, str3); // Append line count

        strcat(strMessage, "\0"); // Null-terminate the message
        printf("\n\tString: %s", strMessage); // Print the final message

        write(fd2, &LC, 1); // Write length of the message to FIFO7
        write(fd2, strMessage, strlen(strMessage)); // Write the message to FIFO7
        fflush(stdin); // Clear the input buffer (not necessary here)

        strMessage[0] = 0; // Reset the character array for the next message

        // Check for termination condition (commented out)
        // if(LC == 1)
        //     break;
    }

    printf("\n"); // Print a newline
    puts("CLIENT CLOSED"); // Indicate the client is closed
    puts("SERVER CLOSED"); // Indicate the server is closed
    close(fd); // Close FIFO6
    close(fd2); // Close FIFO7
```

```
    return 0; // Return success
}
```

Server :

```
#include<stdio.h>       // Include standard input/output library for I/O functions
#include<stdlib.h>      // Include standard library for general functions like memory allocation
#include<unistd.h>      // Include standard symbolic constants and types for UNIX standard
functions
#include<sys/types.h>   // Include definitions for data types used in system calls
#include<fcntl.h>       // Include file control options for file handling
#include<string.h>      // Include string handling functions

int main() {
    int n; // Variable declaration (not used in this snippet)
    puts("Server"); // Print a message indicating that this is the server

    char strMessage[5000]; // Buffer for messages to be sent to the client

    // Open FIFO6 for writing (to send messages to the client)
    int fd = open("fifo6.txt", O_WRONLY);

    // Open FIFO7 for reading (to receive messages from the client)
    int fd2 = open("fifo7.txt", O_RDONLY);

    // Check if opening FIFO6 for writing was successful
    if(fd == -1) {
        perror("cannot open fifo6"); // Print error message if failed
        return EXIT_FAILURE; // Exit the program with failure status
    }

    // Check if opening FIFO7 for reading was successful
    if(fd2 == -1) {
        perror("cannot open fifo7"); // Print error message if failed
        return EXIT_FAILURE; // Exit the program with failure status
    }

    puts("FIFO OPEN"); // Indicate that the FIFOs are successfully open

    // Buffer for reading the incoming message
    char stringBuffer[5000];
    memset(stringBuffer, 0, 5000); // Initialize the buffer to zero

    int res; // Variable for read results (not used in this snippet)
    char Len; // Variable to hold the length of the message

    // Main loop for sending and receiving messages
    {
        // Prompt the user to enter a message
        printf("\n\n\t\tEnter the Message to be passed (hitting ENTER without any string will
terminate program): ");
```

```
        fgets(strMessage, 100, stdin); // Read user input into strMessage

        char L = (char)strlen(strMessage); // Get the length of the input message

        // Write the length of the message to FIFO6
        write(fd, &L, 1);
        // Write the actual message to FIFO6
        write(fd, strMessage, strlen(strMessage));

        fflush(stdin); // Clear the input buffer (not necessary for this use case)

        strMessage[0] = 0; // Reset the character array for the next message

        // Read the length of the response from the client
        int len2;
        res = read(fd2, &len2, 1);

        // Read the actual response message from the client
        read(fd2, stringBuffer, 5000); // Read string characters into the buffer

        // Print the message received from the client
        printf("\nServer Received: %s\n", stringBuffer);
        stringBuffer[(int)len2] = 0; // Null-terminate the received string (this should actually be done
before printing)
    };

    // Cleanup and exit logic (commented out)
    // printf("\n\nCLIENT CLOSED\n")
    // return 0;
}
```

OUTPUT:

CLIENT

```
pl-lab@pllab-OptiPlex-3000:~/client-server$ gcc Client.c
pl-lab@pllab-OptiPlex-3000:~/client-server$ ./a.out

          Client - Listening

FIFO OPEN

Client Received: It's Client-Server Communication...


          String:  No.of Words : 3::: No.of Characters: 35::: No.of Lines: 1
CLIENT CLOSED
SERVER CLOSED
```

SERVER

```
pl-lab@pllab-OptiPlex-3000:~/client-server$ gcc Server.c
pl-lab@pllab-OptiPlex-3000:~/client-server$ ./a.out
Server
```

FIFO OPEN

Enter the Message to be passed (hitting ENTER without any string will terminate program): It's Client-Server Communication...

Server Received: No.of Words : 3::: No.of Characters: 35::: No.of Lines: 1