

TCP Echo Client/Server (Linux) — README

Name: 况旻諭

Student ID: 614430005

1. Overview

This project implements a simple TCP Echo Server and Client using the C language under POSIX/Linux.

- The Server listens on a specified TCP port, accepts incoming client connections, and echoes back each line of text received from the client.
- The Client connects to the server, reads input lines from the user via stdin, sends them to the server, and prints the echoed responses.

Both programs use low-level system calls (`read()`, `write()`, `socket()`, etc.) to ensure compatibility and reliability.

2. Files

- `tcp_server.c` — TCP echo server implementation
 - `tcp_client.c` — TCP echo client implementation
 - `Makefile` — Build script for GNU `make`
 - `614430005_Readme.pdf` — This documentation (exported from Markdown for submission)
-

3. Build Instructions

To compile both the server and client:

```
make
```

To clean up compiled executables:

```
make clean
```

Compiler: `gcc` (or `clang`)

C Standard: `-std=c11`

4. Run Instructions

Run the Server:

```
./tcp_server 5000
```

The server will start listening on port `5000` and display connection logs.

Run the Client:

```
./tcp_client 127.0.0.1 5000
```

Then type a line and press Enter — the server will echo it back.
Press Ctrl + D to exit the client.

5. Implementation Details

Key system calls and their purposes:

Function	Description
<code>socket(AF_INET, SOCK_STREAM, 0)</code>	Creates a TCP socket (IPv4).
<code>setsockopt(..., SO_REUSEADDR, ...)</code>	Allows quick port rebinding after restart.
<code>bind()</code>	Binds the socket to <code>0.0.0.0: <port></code> .
<code>listen()</code>	Marks the socket as a passive listening socket.
<code>accept()</code>	Waits for a client to connect, returns a new connected socket.
<code>connect()</code>	Client connects to server's IP/port.
<code>read() / write()</code>	Handles data transmission at the byte-stream level.
<code>close()</code>	Closes socket and releases system resources.

6. Example Execution

Server Output:

```
[SERVER] Listening on 0.0.0.0: 5000
[SERVER] Connected: 127.0.0.1: 52344
From Client : Hi
[SERVER] Client closed: 127.0.0.1: 52344
```


Client Output:

```
[CLIENT] Connected to 127.0.0.1: 5000
Enter the string : Hi
From Server : Hi
[CLIENT] Input closed.
```

7. Results

 Server-side result:

```
11024102@nhuadmin-WS-E500-G5-WS690T:~/socket-projects/Hw2$ ./tcp_server
[SERVER] Listening on 0.0.0.0:5000
[SERVER] Connected: 127.0.0.1:46092
From Client : hi
█
```

 Client-side result:

```
11024102@nhuadmin-WS-E500-G5-WS690T:~/socket-projects/Hw2$ ./tcp_client
1 5000
[CLIENT] Connected to 127.0.0.1:5000
[CLIENT] Enter the string (Ctrl+D to quit):
Enter str:hi
From Server : hi
Enter str:█
```
