

Project Justification Document

Project Description and Goals

The project aims to implement a file transfer system using the TCP protocol on the application layer. The system consists of a server and a client, allowing them to communicate with each other. The client can either send a file to the server or request a file from the server. The primary goals of the project include:

1. TCP Communication: Utilize TCP sockets for communication between the server and client.
2. File Transfer: Enable the client to send files to the server and request or send files of different types from or to the server.
3. Multiclient Processing: Implement additional features for handling multiple clients concurrently.

Team Contributions

The project development involved the collaboration of two team members, each responsible for specific contributions:

- Changhao Wang:
 - Completed the fundamental structure for file transfer.
 - Implemented the basic file transfer functionality.
- Zijian Fan:
 - Extended the project by adding features such as multiclient processing.
 - Implemented the ability for clients to request files from the server.
 - Implemented the ability for clients to send file to the server
 - Fix some bug

The collaboration ensured a well-rounded implementation, combining foundational structure and additional features.

Instructions for Running and Testing

GitHub link: https://github.com/changhao-wang010220/d58_final_project

To run and test the implementation, follow these instructions:

How to run:

1. Compile the client and server programs:

```
```bash
gcc client.c -o client
gcc server.c -o server
```
```

2. Start the server in one terminal:

```
```bash
./server
```
```

3. Start the client in another terminal:

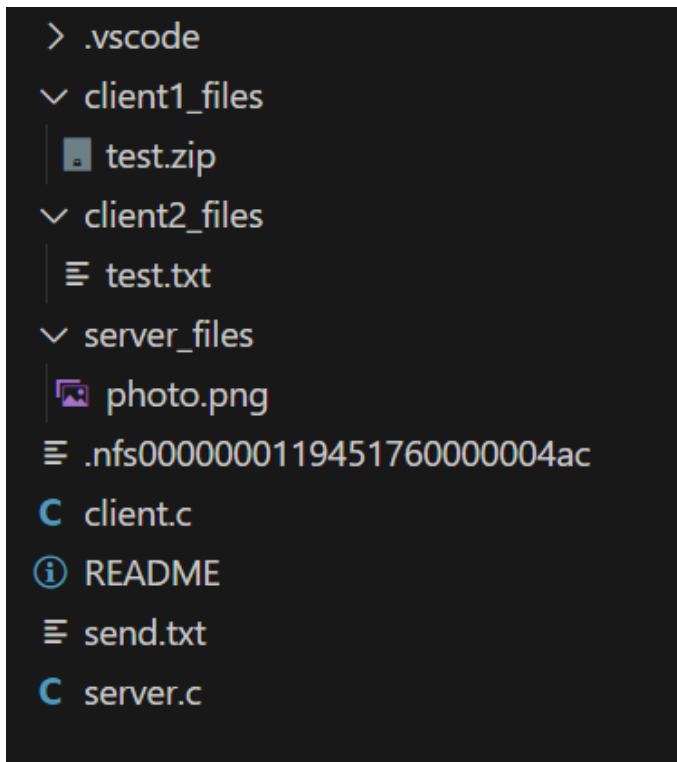
```
```bash
./client (-r or -s) IP
```
```

- Use `-r`` for receiving files from the server.
- Use `-s`` for sending files to the server.
- Replace ``IP`` with the server's IP address (e.g., 127.0.0.7 for local testing).

How to test:

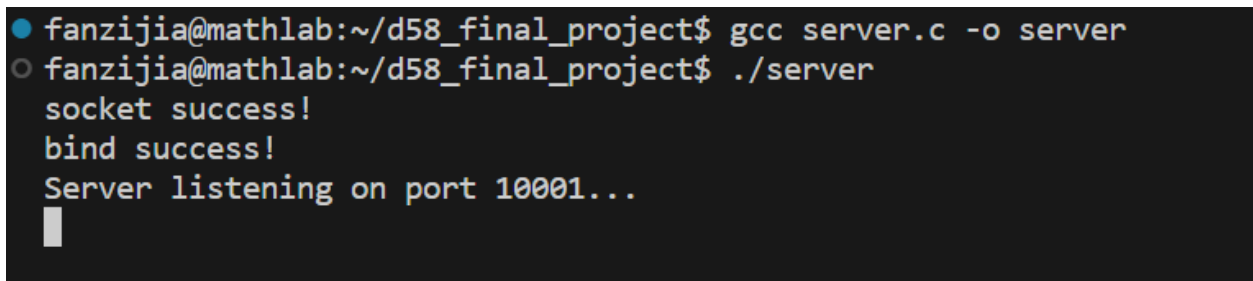
Initial:

The file structure is initially shown as follows.



Server:

We compile the "server.c" file and run it. It will hang as follows.

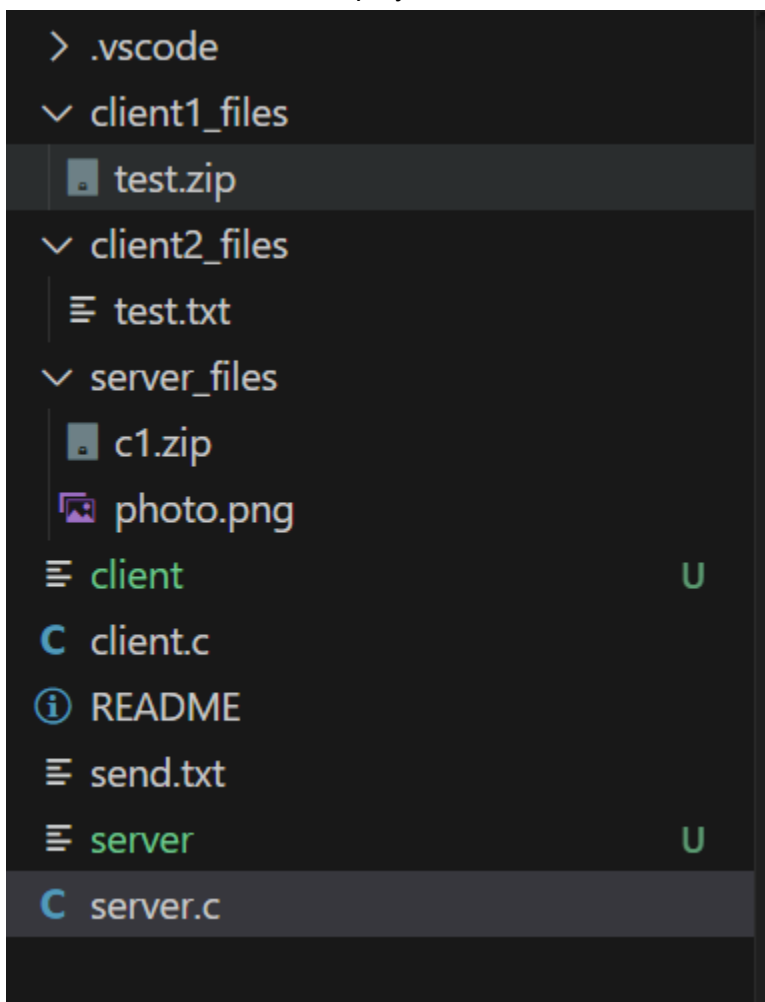


Client1:

For client1, we compile the "client.c" file and run it with the send mode. It will prompt you to enter the source path and store path. After entering "./client1_files/test.zip" as the source path and "c1.zip" as the store path (the path to the server is relative; it is stored to ./server_files/c1.zip), it will be displayed as follows, and the client program will end once it is done.

```
fanzijia@mathlab:~/d58_final_project$ ./client -s 127.0.0.1
-----s-----
socket success!
Enter source path: ./client1_files/test.zip
Enter store path: c1.zip
Received source path: ./client1_files/test.zip
Received store path: c1.zip
Received mode      : s
-----
*****
```

The file structure will be displayed as follows after it's done.

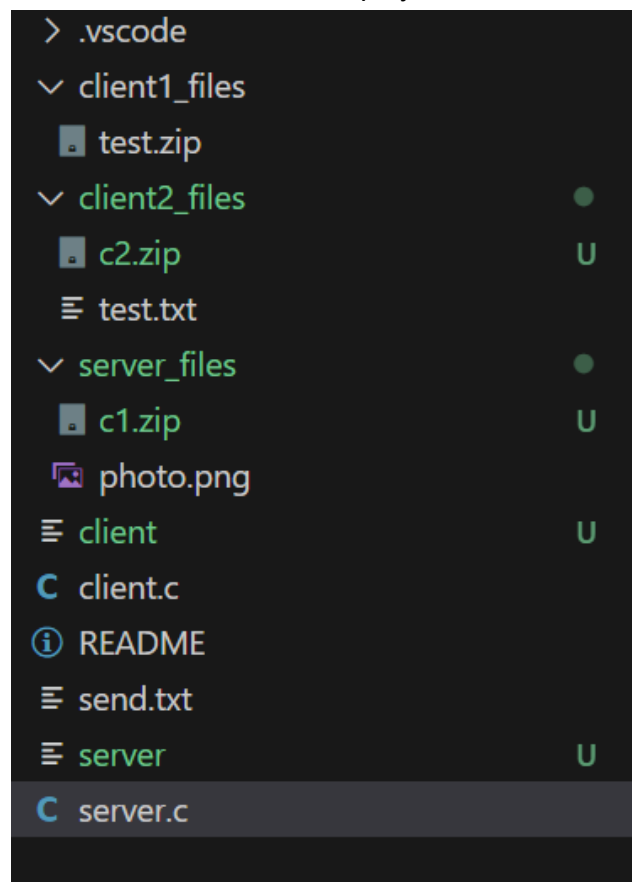


Client2:

For client2, we compile the "client.c" file and run it with the receive mode. It will prompt you to enter the source path and store path. After entering "c1.zip" as the source path and "./client2_files/c2.zip" as the store path (the path to the server is relative; it is received from ./server_files/c1.zip), it will be displayed as follows, and the client program will end once it is done.

```
● fanzijia@mathlab:~/d58_final_project$ ./client -r 127.0.0.1
-----r-----
socket success!
Enter source path: c1.zip
Enter store path: ./client2_files/c2.zip
Received source path: c1.zip
Received store path: ./client2_files/c2.zip
Received mode      : r
file size:3625
*****
```

The file structure will be displayed as follows after it's done.



The screenshot shows a file explorer with the following structure:

- > .vscode
- ▼ client1_files
 - test.zip
- ▼ client2_files
 - c2.zip
 - test.txt
- ▼ server_files
 - c1.zip
 - photo.png
- client
- client.c
- README
- send.txt
- server
- server.c

Result:

As a result, client1 successfully sends test.zip to client2. This program enables different types of file transfer for multiple users. The server will keep hanging for future interaction.

```
fanzijia@mathlab:~/d58_final_project$ ./server
socket success!
bind success!
Server listening on port 10001...
accept success!
Received source path: ./client1_files/test1.zip
Received store path: c1.zip
Received mode      : s
file size error!
accept success!
Received source path: ./client1_files/test.zip
Received store path: c1.zip
Received mode      : s
file size:3625
accept success!
Received source path: c1.zip
Received store path: ./client2_files/c2.zip
Received mode      : r
█
```

Implementation Details and Documentation

- Client (`client.c`)
The client program includes functions for sending file paths to the server, receiving files, and handling different modes (send or receive). It uses TCP sockets for communication.
- Server (`server.c`)
The server program sets up a socket, listens for client connections, and spawns child processes to handle each client. It can receive file paths from clients, send files, and manage multiple client connections concurrently.

Analysis and Discussion

The implementation successfully achieves the project goals, providing a reliable file transfer system over TCP. Multiclient processing ensures the scalability of the system, allowing simultaneous file transfers between multiple clients and the server.

The chosen TCP protocol ensures reliable data delivery, making the system suitable for various applications where data integrity is crucial.

Concluding Remarks and Lessons Learned

In conclusion, the project has provided valuable insights into socket programming, TCP communication, and concurrent processing. The collaboration between team members ensured a comprehensive implementation, combining fundamental functionality with advanced features.

Key lessons learned include effective communication within the team, proper handling of file transfers, and the importance of considering scalability in network applications.

The implemented system can serve as a foundation for further enhancements and applications requiring secure and reliable file transfers. Overall, the project demonstrates the successful application of networking concepts in building a functional file transfer system.