CS339: Computer Networking Assignment 2 Project Report

Weichen Li
5120309662
eizo.lee@sjtu.edu.cn
Department of Computer
Science and Engineering
School of Electronic Information
and Electrical Engineering
Shanghai Jiao Tong University

Abstract

In this project I constructed a FTP Client end system and a FTP Server end system. Client can log in on a SJTU Portal FTP Server or the local FTP Server and it can change directory, list objects, download .txt files, upload .txt files and delete objects on the server side. Server can response to the command coming from Client and perform correspoding actions.

The whole project is pretty like a real FTP Client-Server system, for I implemented the FTP protocol and TCP protocol in a standard way.

CONTENTS

I. SYSTEM DESCRIPTION

A. Function

The system consists of two parts: Client and Server, both of which are controlled in command line. Thus, the only UI in this project is command line.

Client can connect to the Server, or to the SJTU Portal FTP Server, based on the given host name. (**localhost** or **portal.sjtu.edu.cn**, I tested the system on a single computer, so the host name is **localhost**)

Client need to log in on server side. For SJTU Portal FTP Server, the user name and password are corresponding user name and password of JAccount. If the host name of the server is **localhost**, then the user name is ***ComputerNetworking***, and the password is ***123456***.

After logging in, Client is able to

- Change directory.
- List the objects in current directory.
- Download .txt files from server side. (Other types of file are not supported yet)
- Upload .txt files to server side.
- Delete objects on server side.

The commands supported by the system are shown in TABLE I

TABLE I
COMMANDS

Command	Function
PASV	Start the passive mode
PWD	Get the current directory
LIST	List all the files and sub-directories in the current directory
CWD xxxx	Change the current directory to *xxxx*
CDUP	Jump to the upper directory
RETR xxxx	Download file *xxxx*
STOR xxxx	Upload file *xxxx*
QUIT	Close the connection
DELE xxxx	Delete file *xxxx*

B. Runtime Environment Description

The configuration of the environment in which I tested Client and Server is:

- Computer model: Alienware 15
- CPU: Intel Core i7-4710HQ
- Operating system: Windows 8.1 professional
- RAM: 16 GB
- IDE: Visual Studio 2013 Ultimate
- Language: C++

II. SYSTEM DESIGN

With the guidance of the WinSock tutorial, I established the whole project using APIs of WinSock.

A. Architecture

Due to the limited time and resources, I designed the project in a Process-Oriented way. Thus, there is no complex class but simple and effective procedures in my source code.

Here I will only give some abstract description of my source code. For details and practical implements, please refer to the source code.

1) Client: Client mainly consists of three parts: Build control connection to Server, Send user name and password, and Send commands.

- *Build control connection to Server*:
  1) Resolve the server address and port: For SJTU Portal FTP Server, the host name is **portal.sjtu.edu.cn**, the port is **21**. Otherwise, the host name is **localhost**, the port is **21**.
  2) Create a SOCKET for connecting to server: The socket type is SOCK_STREAM. IPv4 is used for IP address. The protocol is TCP.
  3) Connect to Server: Attempt to connect to an address until one succeeds.
  4) Receive message: Receive the response code and welcome message from the server once the connection is established.

- *Send user name and password*: For SJTU Portal FTP Server, the user name and password correspond to the account of JAccount. Otherwise, the user name is **ComputerNetworking**, the password is **123456**.
  1) Send the user name to Server. If there is a corresponding user name on Server, a responsing message will be sent to Client through control connection, with number 331 at the beginning.

2) Send the password to Server. If the password is correct. Server will send a responsing message to Client through control connection, with number 230 at the beginning. And another message will be sent to Client next, with number 230 at the beginning as well.

- *Send commands*: All the commands in this project are listed in TABLE I. And generally a typical procedure is composed of three parts:

1) Client sends a command to Server through control connection.
2) (Optional) Client sends/receives data from Server through data connection. (The data connection is built by another socket instead of the control connection socket. And the data connection must be established before the first step of the procedure. Moreover, we must send a *PASV* command first to establish a passive data connection.)
3) Client receives a response from Server through control connection.

2) Server: Client mainly consists of three parts: Build control connection to Server, Send user name and password, and Send commands.

- *Build control connection*:

1) Resolve the server address and port: The host name is *NULL* (The system will match it with **localhost** automatically). The port is **21**.
2) Create a SOCKET for connecting to server: The socket type is SOCK_STREAM. IPv4 is used for IP address. The protocol is TCP.
3) Setup the TCP listening socket: Bind the socket to the IP address and the port. And listen to the incoming connection from Client.
4) Accept Client socket: Obtain the client socket. The connection is established.

- *Receive user name and password*: The user name is **ComputerNetworking**, the password is **123456**.

1) Receive the user name from Client. If the corresponding user name is correct, send a responsing message to Client through control connection, with number 331 at the beginning.
2) Receive the password from Client. If the password is correct, send a responsing message to Client through control connection, with number 230 at the beginning. And send another message to Client next, with number 230 at the beginning as well.

- *Receive commands*: This part has been described in Client part already.

B. UI

As I mentioned, the whole projcet is based on command line controlling, so the UI is kind of simple with a command line window.

We mainly focus on the UI of Client. Once the Client logs in on Server, there will be some welcome messages and confirming messages shown in the command line window. And each time we send a command to Server, corresponding messages will shown in the window as well. (As shown in Fig 1)

C. Process Diagram

The process diagram is shown in Fig 2.

D. Files Description

Since I use Visual Studio 2013 Ultimate as my IDE, there is a .sln file, a .sdf file and two directories: Debug directory contains the .exe file, .ilk file and .pdb file. Client(Server) directory contains the main.cpp, .filters file, .vcxproj file, .idc file and another Debug directory (which is not essential so we ignore it).

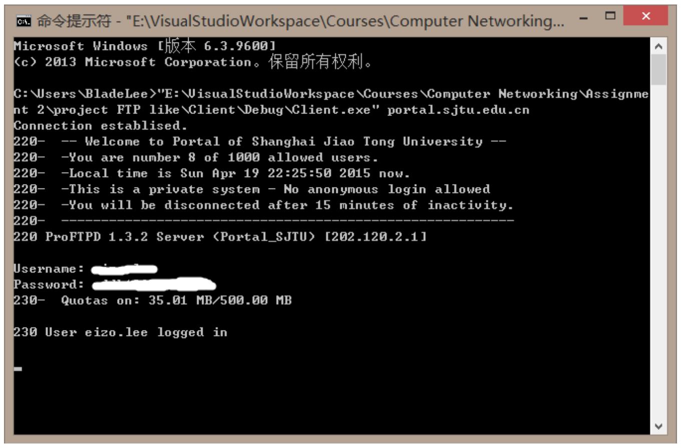
E. Security Feature

The most important part of the project regarding to security may be the sending procedure of user name and password. I send them in plain text. This may lead to some security issues. However, regarding to the scale and requests of this project, I think complex security methods are not necessary.

III. Build and Run

To make things clear, I will use figures to illustrate the process of building and running.

There are two project in my Visual Studio working space, one is Client, the other is Server.


命令提示符 - "E:\VisualStudioWorkspace\Courses\Computer Networking...

```
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\BladeLee>"E:\VisualStudioWorkspace\Courses\Computer Networking\Assignme
nt 2\project FTP like\Client\Debug\Client.exe" portal.sjtu.edu.cn
Connection establised.
220-   -- Welcome to Portal of Shanghai Jiao Tong University --
220-   -You are number 8 of 1000 allowed users.
220-   -Local time is Sun Apr 19 22:25:50 2015 now.
220-   -This is a private system - No anonymous login allowed
220-   -You will be disconnected after 15 minutes of inactivity.
220-   ---------------------------------------------------------
220 ProFTPD 1.3.2 Server (Portal_SJTU) [202.120.2.1]

Username: [redacted]
Password: [redacted]
230-   Quotas on: 35.01 MB/500.00 MB

230 User eizo.lee logged in
```

Fig. 1. UI example

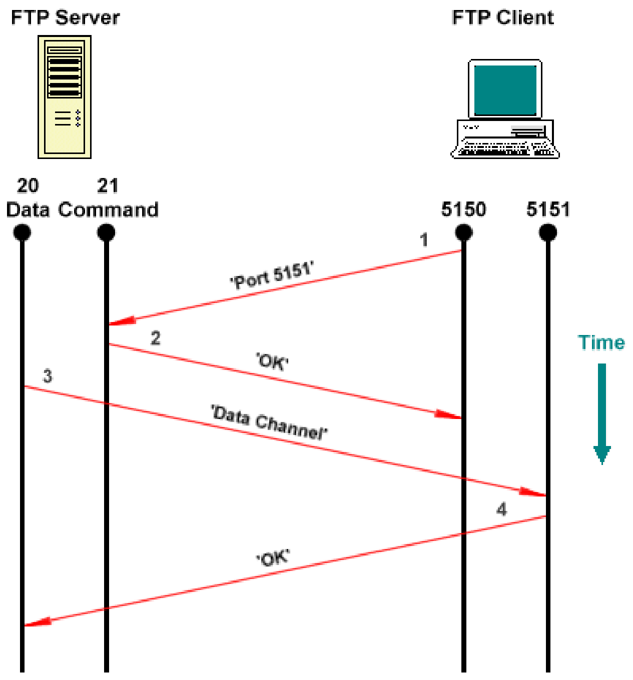


Fig. 2. Process diagram example

A. *Build Method*

1) Client: In Client project, press ”Ctrl + F5” to compile and build the source code. A window like Fig 3 will show. The message in the window tells us that we need to use command line to start Client, with one argument indicating the host name of the server.

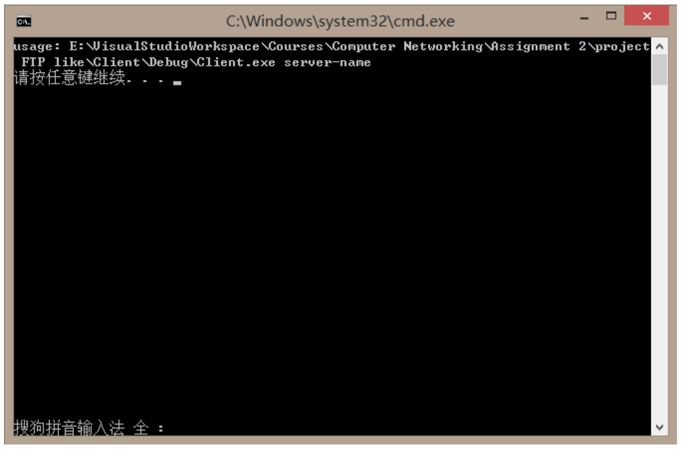
2) Server: In Server project, just press ”Ctrl + F5” and Server will activate automatically, waiting for Client to connect.

B. *Start and Run*

We will only introduce Client here, since Server is very easy to start: Just open Server.exe file (We will use both Client and Server when we build a local FTP connection, but when we want to connect to SJTU Portal FTP Server, we need only use Client).

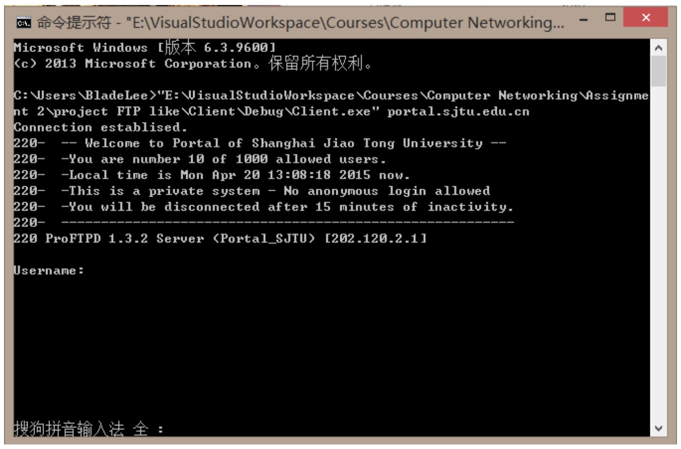
We need to use CMD to start Client.exe, with one argument **portal.sjtu.edu.cn** (Or **localhost** in local FTP connection mode).

- *Connection established*: We can receive some messages like Fig 4.
- *Input user name and password*: If we are building local FTP connection, the user name is **ComputerNetworking**, the password is **123456**. If we are connecting SJTU Portal FTP Server, we need to use JAccount. Then we can receive



```
usage: E:\VisualStudioWorkspace\Courses\Computer Networking\Assignment 2\project
 FTP like\Client\Debug\Client.exe server-name
请按任意键继续. . .
```

Fig. 3. Incorrect Startup

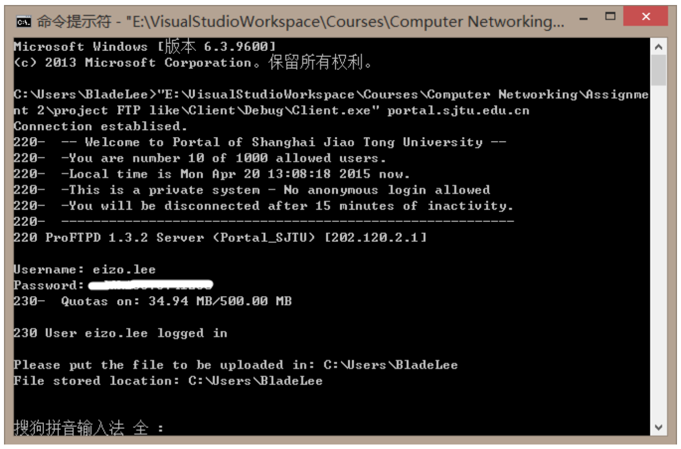


```
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\BladeLee>"E:\VisualStudioWorkspace\Courses\Computer Networking\Assignme
nt 2\project FTP like\Client\Debug\Client.exe" portal.sjtu.edu.cn
Connection established.
220-  -- Welcome to Portal of Shanghai Jiao Tong University --
220-  -You are number 10 of 1000 allowed users.
220-  -Local time is Mon Apr 20 13:08:18 2015 now.
220-  -This is a private system - No anonymous login allowed
220-  -You will be disconnected after 15 minutes of inactivity.
220-  ---------------------------------------------------------
220 ProFTPD 1.3.2 Server (Portal_SJTU) [202.120.2.1]

Username:
```

命令提示符 - "E:\VisualStudioWorkspace\Courses\Computer Networking...
搜狗拼音输入法 全 :

Fig. 4. Correct Startup

confirming messages like Fig 5 (I used JAccount to illustrate). Notice that the window will tell us about the directory in which we must put the file to be upoladed, or the file will be downloaded.

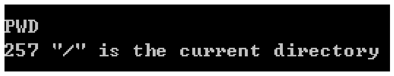


```
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\BladeLee>"E:\VisualStudioWorkspace\Courses\Computer Networking\Assignme
nt 2\project FTP like\Client\Debug\Client.exe" portal.sjtu.edu.cn
Connection established.
220-  -- Welcome to Portal of Shanghai Jiao Tong University --
220-  -You are number 10 of 1000 allowed users.
220-  -Local time is Mon Apr 20 13:08:18 2015 now.
220-  -This is a private system - No anonymous login allowed
220-  -You will be disconnected after 15 minutes of inactivity.
220-  ---------------------------------------------------------
220 ProFTPD 1.3.2 Server (Portal_SJTU) [202.120.2.1]

Username: eizo.lee
Password:[redacted]
230-  Quotas on: 34.94 MB/500.00 MB

230 User eizo.lee logged in

Please put the file to be uploaded in: C:\Users\BladeLee
File stored location: C:\Users\BladeLee
```

命令提示符 - "E:\VisualStudioWorkspace\Courses\Computer Networking...
搜狗拼音输入法 全 :

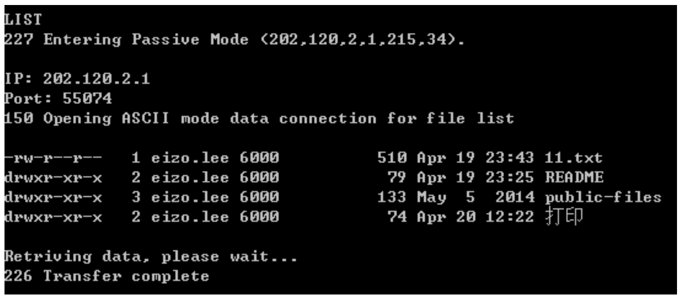
Fig. 5. User name and password confirmed

- *PWD command*: Shown in Fig 6.
- *LIST command*: Shown in Fig 7.
- *CWD command*: Shown in Fig 8.



```
PWD
257 "/" is the current directory
```

Fig. 6. PWD command



```
LIST
227 Entering Passive Mode (202,120,2,1,215,34).

IP: 202.120.2.1
Port: 55074
150 Opening ASCII mode data connection for file list

-rw-r--r--   1 eizo.lee 6000         510 Apr 19 23:43 11.txt
drwxr-xr-x   2 eizo.lee 6000          79 Apr 19 23:25 README
drwxr-xr-x   3 eizo.lee 6000         133 May  5  2014 public-files
drwxr-xr-x   2 eizo.lee 6000          74 Apr 20 12:22 打印

Retriving data, please wait...
226 Transfer complete
```

Fig. 7. LIST command

- *CDUP command*: Shown in Fig 9.
- *RETR command*: Shown in Fig 10.
- *STOR command*: Shown in Fig 11.
- *DELE command*: Shown in Fig 12.
- *QUIT command*: Shown in Fig 13.

IV. Problems and Experiences

Through out the whole procedure of designing and implementing of the system, I met several problems using Winsock APIs:

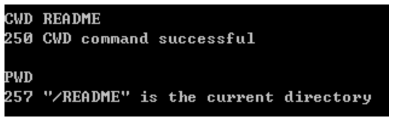
- How to build a control connection?
- How a command is realized?
- How to build a data connection?
- How to send/receive a file via data connection?

A. Troubleshooting Experiences

For each of the problem above, I will illustrate my relevant experience from the perspectives of both Client and Server.

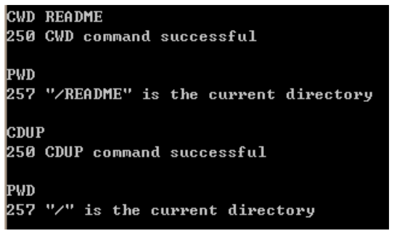
1) How to build a control connection:

- *Client*: Use function *getaddrinfo* to resolve the address and port of Server. Create a socketA using function *socket*. Connect socketA and Server using function *connect*.
- *Server*: Use function *getaddrinfo* to resolve the address and port of Server. Create a socketB using function *socket*. Use *bind* and *listen* to setup the TCP listening socket. Create a socketC, which is the accepted Client socket coming from function *accept*.



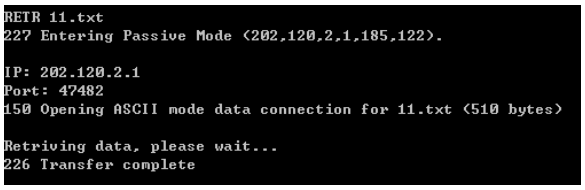
```
CWD README
250 CWD command successful

PWD
257 "/README" is the current directory
```

Fig. 8. CWD command



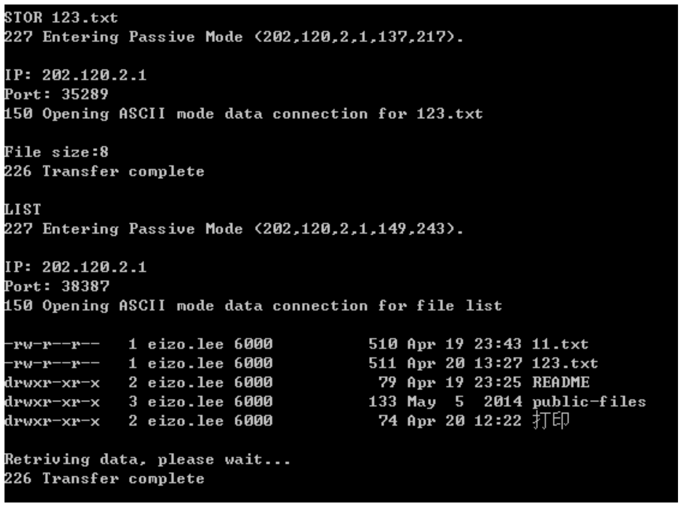
```
CWD README
250 CWD command successful

PWD
257 "/README" is the current directory

CDUP
250 CDUP command successful

PWD
257 "/" is the current directory
```

Fig. 9. CDUP command



```
RETR 11.txt
227 Entering Passive Mode (202,120,2,1,185,122).

IP: 202.120.2.1
Port: 47482
150 Opening ASCII mode data connection for 11.txt (510 bytes)

Retriving data, please wait...
226 Transfer complete
```

Fig. 10. RETR command



```
STOR 123.txt
227 Entering Passive Mode (202,120,2,1,137,217).

IP: 202.120.2.1
Port: 35289
150 Opening ASCII mode data connection for 123.txt

File size:8
226 Transfer complete

LIST
227 Entering Passive Mode (202,120,2,1,149,243).

IP: 202.120.2.1
Port: 38387
150 Opening ASCII mode data connection for file list

-rw-r--r--   1 eizo.lee 6000          510 Apr 19 23:43 11.txt
-rw-r--r--   1 eizo.lee 6000          511 Apr 20 13:27 123.txt
drwxr-xr-x   2 eizo.lee 6000           79 Apr 19 23:25 README
drwxr-xr-x   3 eizo.lee 6000          133 May  5  2014 public-files
drwxr-xr-x   2 eizo.lee 6000           74 Apr 20 12:22 打印

Retriving data, please wait...
226 Transfer complete
```

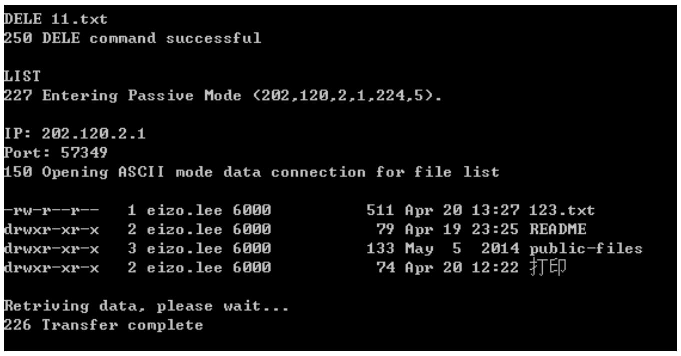
Fig. 11. STOR command

2) How a command is realized:

- *Client*: Use function *send* to send a command to Server, and use function *recv* receive respond messages from Server.
- *Server*: Wait for incoming messages from Client using *recv* function, and use *send* to send response message back to Client.

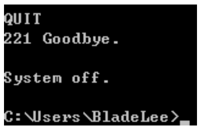
3) How to build a data connection:

- *Client*: Send a *PASV* command to Server first, receive corresponding responding message. Then use function *getaddrinfo* to resolve the address and port of Server. Create a socketC using function *socket*. Connect socketC and Server using function *connect*.



```
DELE 11.txt
250 DELE command successful

LIST
227 Entering Passive Mode (202,120,2,1,224,5).

IP: 202.120.2.1
Port: 57349
150 Opening ASCII mode data connection for file list

-rw-r--r--   1 eizo.lee 6000          511 Apr 20 13:27 123.txt
drwxr-xr-x   2 eizo.lee 6000           79 Apr 19 23:25 README
drwxr-xr-x   3 eizo.lee 6000          133 May  5  2014 public-files
drwxr-xr-x   2 eizo.lee 6000           74 Apr 20 12:22 打印

Retriving data, please wait...
226 Transfer complete
```

Fig. 12. DELE command



```
QUIT
221 Goodbye.

System off.

C:\Users\BladeLee>
```

Fig. 13. QUIT command

- *Server*: Upon receiving a *PASV* command, send back a responsing message to Client. Then use function *getaddrinfo* to resolve the address and port of Server. Create a socketB using function *socket*. Use *bind* and *listen* to setup the TCP listening socket. Create a socketD, which is the accepted Client socket coming from function *accept*.

4) How to send/receive a file via data connection:

- *Client*: For command *RETR* and *STOR* (*LIST* as well), we need to use data connection to transmit data. For *RETR*, we send the *RETR* command first, then we will receive a message from Server, indicating the size of the file and the status of Server (Ready to send data to Client). Then we use *recv* on socketC to receive data, until the connection is closed by Server, which means that connection is complete. Of course, finally there will be a responding message on control connection to notify the transmission is completed. For *STOR*, we send the *STOR* command first, then we will receive a message from Server, indicating the status of Server (Ready to receive data from Client). Then we use *send* on socketC to send data. When the data is transmitted, use *shutdown* to close socketC, this will tell the Server that Client has completed the transmission. Finally Server will send a message to Client notifying that transmission is completed. Of course, finally there will be a responding message on control connection to notify the transmission is completed.
- *Server*: Generally, the actions Server will take is the inverse of Client. Thus, for details please go to the source code.

B. Comments and Suggestion

This project is pretty challenging for me, since I have no network programming experience before. I spent a lot of time searching for guidances and examples on the Internet. However, few resources can be referred to.

I hope that if we could have more study resources given by teacher or TA, I will complete the project faster and add more functions in it.