Project report

Design and implementation of FTP Client

Course Title: *Internet Application*

Name: *Wang Xutao(2016213034)*

Wang Zengze (2016213065)

Date: 2019/6/4

1. Overview

The FTP Client lab is mainly about completing a FTP client program based on Linux command, which aims at helping us understand FTP. FTP is File Transfer Protocol, which can be used to transfer files between hosts and manipulate files. FTP will make two connections: control connection and data connection. What's more, data connection has two modes: active mode and passive mode. This FTP client will be able to login FTP server and open control connection and data connection in passive mode and active mode. In addition, it can apply communication procedure, commands and relies of FTP.

2. Requirements Analysis

Basic Requirements of the project

The FTP client should be written by C language in Linux environment.

Basic requirements:

1. The well-known port number (TCP Port 21) of FTP server is used and there is no need to input it. The ftp server can be identified by IP address, such as: ./ftpcli 10.3.255.85 ./ftpcli 127.0.0.1

   Analysis: The first step is to establish a TCP connection from client to server, defining function cliopen(), it is passive mode. Function connect() is used in Linux to make connection from client to server. Port number 21 should be set in main connection to make connection.

2. FTP client connects FTP server using TCP protocol. It can receive the FTP replies coming from server and translate it into natural language, and change the user's commands into FTP command before sending.

   Analysis: we use function FD_ISSET() to receive data from socket and translate them into nature language. We use function strncmp() to check the input command by user. It should be noticed that the number of data bytes should be set before read standard input.

3. Support user's authentication by username & password, with hidden password function.

   Analysis: The input user name should be firstly assigned to name and then checked. First write into socket and then wait and read the reply from server. After passing, check password. Function system("stty -echo") is used to hide the password. After successfully input password, remember to return to normal mode.

4. Support user's interactive operation and provide the following commands: list (ls), directory operation (pwd, cd, mkdir), upload a file (put), download a file (get) , delete a file (delete), renaming a file (rename), transfer mode (ascii, binary), and exit (quit);

   Analysis: Firstly, after the user's state is set to login, then commands can be listened. If the user input ls command, the client write this command into packet that will be sent to the server and so on. It should be mentioned that passive mode and active mode should be selected.

5. Be able to handle errors: invalid commands, missing parameters, requested file already existed.

   Analysis: Invalid input should be checked firstly and it should be sure that the error ftp server reply should be received by client.

6. Detailed designing document and user manual.
7. Detailed annotation of code and nice programming style.

8. Stable and friendly to users.
9. Two persons as a group.

Extension requirements

1. Active mode for data connection;

   Analysis: Active mode could be set as a command operation.
2. Resuming transmission from break-point;

   Analysis: When a file is resumed, then the existing file could not be empty. We use function lseek() to check the existing file. If it is not empty, then the main loop will restart, which means beginning the second transmission. Input command could be used to active the second transmission.
3. Limiting transmission data rate;

   Analysis: Set a intergrate number to represent the state of speed and we decide to Use function sleep() to slow down the transmission rate. The operation can be input as command.

3. Preliminary Design

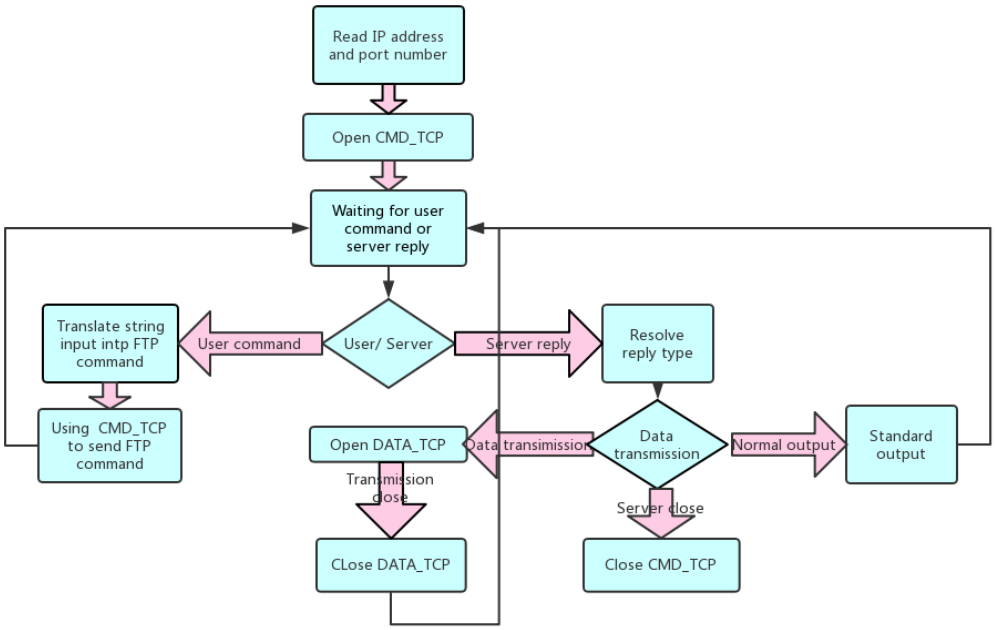
3.1 Decomposition of functional modules

We divide the code into three parts: make connection, main loop and basic functions. The first part is making connection. There are two mode to be chosen: active mode and passive mode. The normal mode is passive and active mode can be stimulate by inputting command. The second part is main loop, a for loop will be used to read and run the input command from client. The last part is functions, including the function for read and write in connection, the function used for downloading file from ftp server and the function used for upload the file to ftp server.

3.2 Relationship and interface between the modules

Making connection module is related to main loop. After successfully login, a socket will be open and then the main loop will be used to read from client and run input command. The main loop is related with basic function module. Different input command will run according functions.

3.3 Overall flow chart



4. Detailed Design

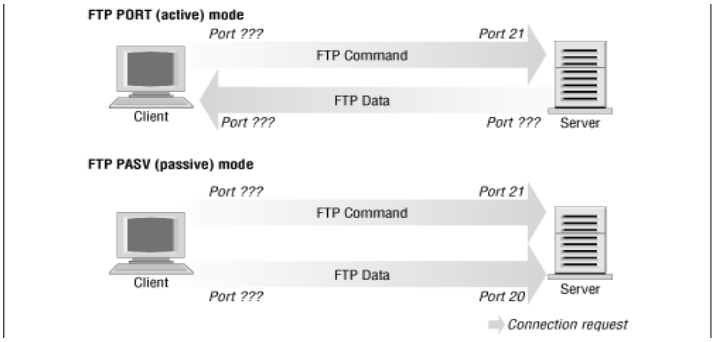
4.1 Design analysis of each module

4.1.1 `int cliopen(char *host, char *port)`

*Function cliopen (char *host, char *port) is used to establish a TCP from client to server. Function socket(PF_INET, SOCK_STREAM, 0) is used to open a TCP socket and then initialize sockaddr_in structure, normally address family is assigned to AF_INET, which means using IP address. Input IP host address is assigned to host, and then function connect() is used to establish the connection to assigned server.*

4.1.2 `ftp_activeopen(int port1)`

Function ftp_activeopen is the function that used to establish active mode TCP connection. In active mode, FTP client does not establish a connection, but reply its listing port number to server, then the server will make connection with the assigned port number. For client's firewall, this is actually an outside connection that led to inside client, so this kind of connection will usually be blocked.



The design code of active mode is divided into 4 steps. The first step is open a socket and then initialize it with assigned IP address and port number. The next step is using function bind() to assign the client address to socket. The third step is using function listen() to make socket listen from server. The last step is using function accept() to wait and reply the query to establish a new socket.

4.1.3 `void strtosrv(char *str, char *host, char *port)`
Function strtosrv() is used to attract the host address and calculate port number.

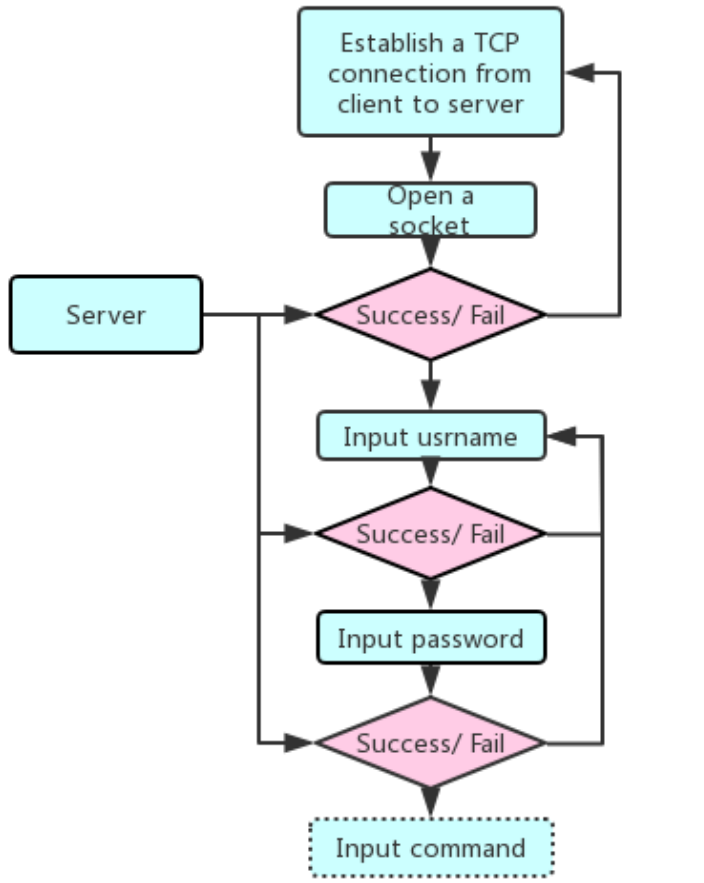
4.1.4 `void cmd_tcp(int sockfd)`
Function cmd_tcp as the the main loop that is used to read the command from user input and then translate them into TCP command language in control connection. Infinite loop for is used to repeat the request. Function select is used to monitor file descriptors, thus to handle I/O operations.
Integrate number replycode is used to identify the state of user.
Function FD_ISSET is used to read data from user input. "ls" is the command that list files and directories; "put " is used to upload a file; "get" is used to download a file. "dele" is used to delete a file. "rename" is used to rename a file. "ascii" and "binary" is used to transfer mode between these two types. "quit" is used to exit the system. "pwd" is used to print working directory. "pwd" is used to return back to main directory. "cwd" is used to transfer to a directory. "mkd" is used to make a directory. Function FD_ISSET is used to judge whether the input is from client or sever. After receive the reply code from server, check the number according to different control commands. And then wait for the user input.

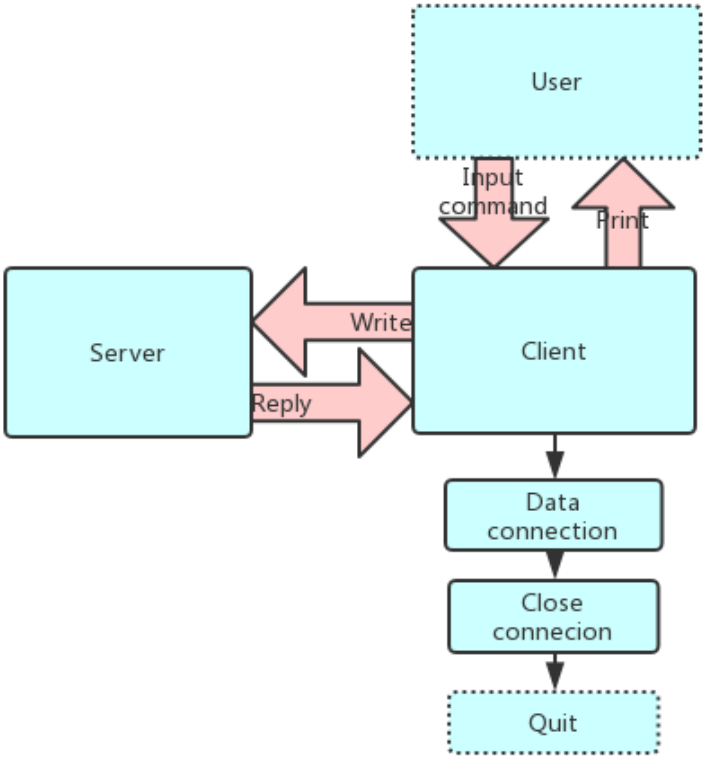
4.1.5 `void ftp_list(int sockfd)`
Function ftp_list is used to read and write in data connection, list the directory.

4.1.6 `int ftp_get(int sck,char *pDownloadFileName)`
Function ftp_get is used to download file from ftp server. And break-point transmission and limitation of transmission speed function are set in this part. When a file is resumed, then the existing file could not be empty. We use function lseek() to check the buffer. If it is not empty, then the main loop will restart, which means beginning the second transmission. Input command could be used to active the second transmission. Set a intergrate number to represent the state of speed and we decide to Use function sleep() to slow down the transmission rate. The operation can be input as command.

4.1.7 `int ftp_put (int sck, char *pUploadFileName_s)`
Function ftp_put is used to upload file to ftp server.

4.2 Flow chart of each module

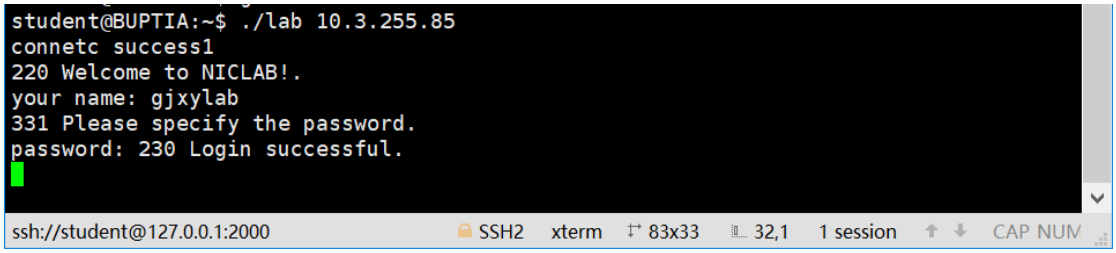

Establish a TCP connection from client to server
Open a socket
Server
Success/ Fail
Input usrname
Success/ Fail
Input password
Success/ Fail
Input command


User
Input command
Print
Server
Write
Reply
Client
Data connection
Close connecion
Quit

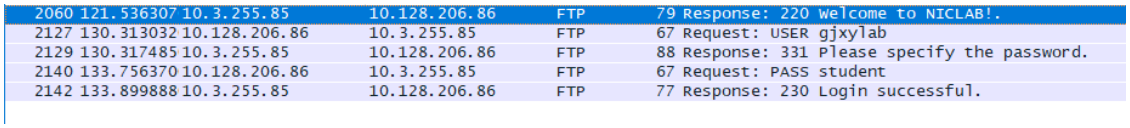
5. Results

5.1 Make connection and login

Xshell:

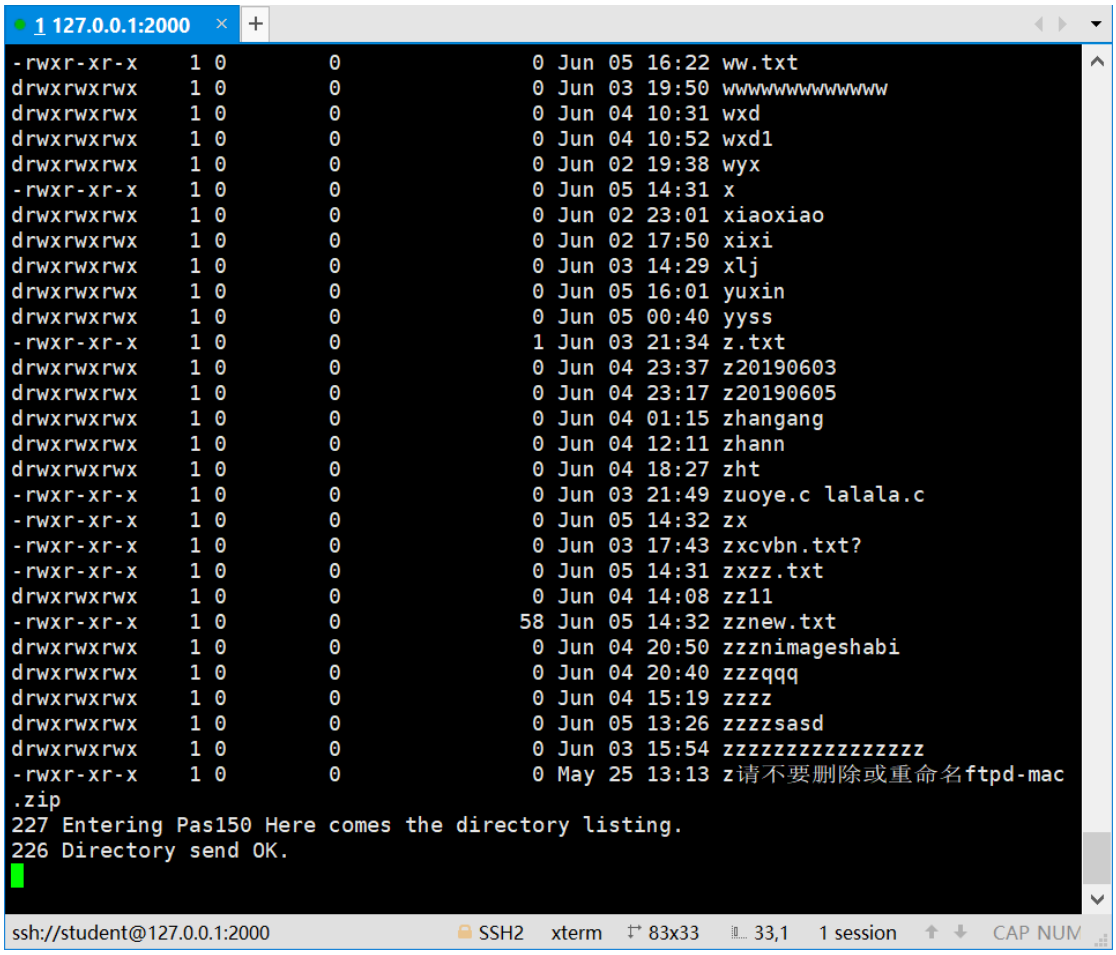


Wireshark:

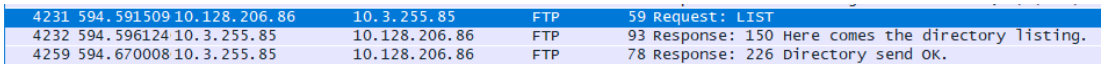


5.2 User command

Xshell: ls



Wireshark:



Xshell:

```
pwd /
257 "/" is the current directory
mkd
550 Create directory operation failed.
pwd
257 "/" is the current directory
mkd root
257 "/root" created
pwd
257 "/" is the current directory
cmd root

cwd root
250 Directory successfully changed.
```

ssh://student@127.0.0.1:2000 SSH2 xterm 83x33 33,1 1 session CAP NUM

Wireahrk:

9971	991.073467	10.128.206.86	10.3.255.85	FTP	58 Request: PWD
9972	991.077213	10.3.255.85	10.128.206.86	FTP	88 Response: 257 "/" is the current directory
10002	998.977873	10.128.206.86	10.3.255.85	FTP	58 Request: mkd
10003	998.982652	10.3.255.85	10.128.206.86	FTP	94 Response: 550 Create directory operation failed.
11031	1055.95212	10.128.206.86	10.3.255.85	FTP	58 Request: PWD
11032	1055.95821	10.3.255.85	10.128.206.86	FTP	88 Response: 257 "/" is the current directory
11093	1070.31509	10.128.206.86	10.3.255.85	FTP	63 Request: mkd root
11094	1070.32202	10.3.255.85	10.128.206.86	FTP	75 Response: 257 "/root" created
11214	1082.06489	10.128.206.86	10.3.255.85	FTP	58 Request: PWD
11215	1082.07110	10.3.255.85	10.128.206.86	FTP	88 Response: 257 "/" is the current directory
11599	1152.56516	10.128.206.86	10.3.255.85	FTP	63 Request: cwd root
11600	1152.57031	10.3.255.85	10.128.206.86	FTP	91 Response: 250 Directory successfully changed.

Xhsell:

```
wqd
wqd
wq
d
wq227 Entering PasSTOR 44.txt
,3,q
d
wqd
wqd
w150 Ok to send data.
226 Transfer complete.
get 44.txt
44.txt
connetc success1
RETR 44.txt

12
6
over
227 Entering PasRETR 44.txt
,3,q
d
wqd
wqd
150 Opening ASCII mode data connection for 44.txt (1278 bytes).
226 Transfer complete.
rename 44.txt
350 Ready for RNTO.
new name: 22.txt
250 Rename successful.
dele 22.txt
250 Delete operation successful.
```

ssh://student@127.0.0.1:2000 SSH2 xterm 83x33 33,1 1 session CAP NUM

Wireshark:

```
13501 1383.35543 10.128.206.86   10.3.255.85      FTP   66 Request: STOR 44.txt
13507 1383.35978 10.3.255.85     10.128.206.86    FTP   76 Response: 150 Ok to send data.
13510 1383.36314 10.3.255.85     10.128.206.86    FTP   78 Response: 226 Transfer complete.
13633 1415.02381 10.128.206.86   10.3.255.85      FTP   59 Request: PASV
13634 1415.03010 10.3.255.85     10.128.206.86    FTP  103 Response: 227 Entering Passive Mode (10,3,255,85,173,55).
13638 1415.03534 10.128.206.86   10.3.255.85      FTP   66 Request: RETR 44.txt
13639 1415.03890 10.3.255.85     10.128.206.86    FTP  119 Response: 150 Opening ASCII mode data connection for 44.txt (1278 bytes).
13645 1415.04108 10.3.255.85     10.128.206.86    FTP   78 Response: 226 Transfer complete.
14605 1478.10783 10.128.206.86   10.3.255.85      FTP   66 Request: RNFR 44.txt
14609 1478.11195 10.3.255.85     10.128.206.86    FTP   75 Response: 350 Ready for RNTO.
15200 1494.27382 10.128.206.86   10.3.255.85      FTP   66 Request: RNTO 22.txt
15201 1494.27796 10.3.255.85     10.128.206.86    FTP   78 Response: 250 Rename successful.
15250 1498.31441 10.128.206.86   10.3.255.85      FTP   66 Request: dele 22.txt
15251 1498.31918 10.3.255.85     10.128.206.86    FTP   88 Response: 250 Delete operation successful.
```

Xshell:

```
ascii
200 Switching to ASCII mode.
binary
200 Switching to Binary mode.
quit
221 Goodbye.
student@BUPTIA:~$
```

ssh://student@127.0.0.1:2000 SSH2 xterm 83x33 33,19 1 session CAP NUM

Wireshark:

```
15920 1623.73594 10.128.206.86   10.3.255.85      FTP   61 Request: TYPE A
15921 1623.74600 10.3.255.85     10.128.206.86    FTP   84 Response: 200 Switching to ASCII mode.
15938 1630.62744 10.128.206.86   10.3.255.85      FTP   61 Request: TYPE I
15939 1630.63440 10.3.255.85     10.128.206.86    FTP   85 Response: 200 Switching to Binary mode.
15949 1633.91991 10.128.206.86   10.3.255.85      FTP   59 Request: QUIT
15950 1633.92373 10.3.255.85     10.128.206.86    FTP   68 Response: 221 Goodbye.
```

Xshell:

```
lowspeed
lowspeed
get 22.txt
22.txt
connetc success1
RETR 22.txt

12
5
```

ssh://student@127.0.0.1:2000 SSH2 xterm 83x35 35,1 1 session CAP NUM

Wireshark:

```
783 168.812210 10.3.255.85       10.128.206.86    FTP   79 Response: 220 Welcome to NICLAB!.
818 173.552009 10.128.206.86     10.3.255.85      FTP   67 Request: USER gjxylab
820 173.560382 10.3.255.85       10.128.206.86    FTP   88 Response: 331 Please specify the password.
846 176.041936 10.128.206.86     10.3.255.85      FTP   67 Request: PASS student
849 176.281317 10.3.255.85       10.128.206.86    FTP   77 Response: 230 Login successful.
876 187.311352 10.128.206.86     10.3.255.85      FTP   59 Request: PASV
878 187.319291 10.3.255.85       10.128.206.86    FTP  104 Response: 227 Entering Passive Mode (10,3,255,85,190,204).
882 187.329371 10.128.206.86     10.3.255.85      FTP   66 Request: RETR 22.txt
883 187.338528 10.3.255.85       10.128.206.86    FTP   80 Response: 550 Failed to open file.
```

nc command and active mode

Xshell:

```
Connection to ftp.mayan.cn 21 port [tcp/ftp] succeeded!
220 Welcome to NICLAB!.
USER gjxylab
331 Please specify the password.
PASS student
230 Login successful.
PORT 10,128,226,101,10,0
200 PORT command successful. Consider using PASV.
LIST
150 Here comes the directory listing.
226 Directory send OK.
```

```
drwxrwxrwx    1 0        0               0 Jun 06 02:10 what
-rwxr-xr-x    1 0        0              35 Jun 06 09:03 wwaa.txt
drwxrwxrwx    1 0        0               0 Jun 06 08:30 wwwww
-rwxr-xr-x    1 0        0             223 Jun 06 08:34 xiwanying.t
xt
drwxrwxrwx    1 0        0               0 Jun 05 21:10 xxx
drwxrwxrwx    1 0        0               0 Jun 05 21:14 xxyy
drwxrwxrwx    1 0        0               0 Jun 06 08:11 yang
-rwxr-xr-x    1 0        0               0 Jun 06 09:04 year.txt
-rwxr-xr-x    1 0        0           23075 Jun 06 08:51 yes.png
-rwxr-xr-x    1 0        0               1 Jun 03 21:34 z.txt
-rwxr-xr-x    1 0        0               0 Jun 05 20:04 zdjsiofjosi
ao.txt
drwxrwxrwx    1 0        0               0 Jun 05 21:38 zxcv
drwxrwxrwx    1 0        0               0 Jun 06 05:21 zyp
drwxrwxrwx    1 0        0               0 Jun 06 01:14 zzds
drwxrwxrwx    1 0        0               0 Jun 06 09:01 zzza
-rwxr-xr-x    1 0        0               0 May 25 13:13 z请不要删除
或重命名ftpd-mac.zip
-rwxr-xr-x    1 0        0               0 Jun 05 20:00 你好。txt
student@BUPTIA:~$
```

Wireshark:

1773	34.743163000	10.3.255.85	10.0.2.15	FTP	79	Response: 220 Welcome to NICLAB!.
1910	39.275644000	10.0.2.15	10.3.255.85	FTP	67	Request: USER gjxylab
1914	39.286746000	10.3.255.85	10.0.2.15	FTP	88	Response: 331 Please specify the password.
2017	43.190377000	10.0.2.15	10.3.255.85	FTP	67	Request: PASS student
2021	43.327628000	10.3.255.85	10.0.2.15	FTP	77	Response: 230 Login successful.
2468	77.792428000	10.0.2.15	10.3.255.85	FTP	79	Request: PORT 10,128,226,101,10,0
2472	77.807077000	10.3.255.85	10.0.2.15	FTP	105	Response: 200 PORT command successful. Consider using PASV.
2756	95.549209000	10.0.2.15	10.3.255.85	FTP	59	Request: LIST
2763	95.560323000	10.3.255.85	10.0.2.15	FTP	93	Response: 150 Here comes the directory listing.
2822	95.615485000	10.3.255.85	10.0.2.15	FTP	78	Response: 226 Directory send OK.

6. Summary and Conclusion

Wang Xutao is responsible for design the code structure and modules of project, write the main loop, read and write function, download file function and upload file function and some part of report.

Wang Zengze is responsible for requirement analysis of project and write the establish connection function, calculate port number function and some basic function as user command and major part of report.

We know FTP is a standard TCP based network protocol used to transfer files from one host to another host. And it's a client-server architecture. FTP programs were based on command-lines. In order to run the commands that input from client, the FTP server needs to be running and waiting for incoming requests.

Wikipedia explains FTP as "The client computer is able to communicate with the server on port 21 and it's called the control connection. It remains open for the duration of the session, with a second connection, called the data connection either opened by the server from its port 20 to a negotiated client port (active mode) or opened by the client from an arbitrary port to a negotiated server port (passive mode) as required to transfer file data."

From this project, we know and understand the insight substructure of FTP.