Project report

Design and implementation of FTP Client

Course Title: Internet Application
Jincheng Liu(2016213240)
Linger Shen (2016213272)
Date: 2019/6/7

1. Overview

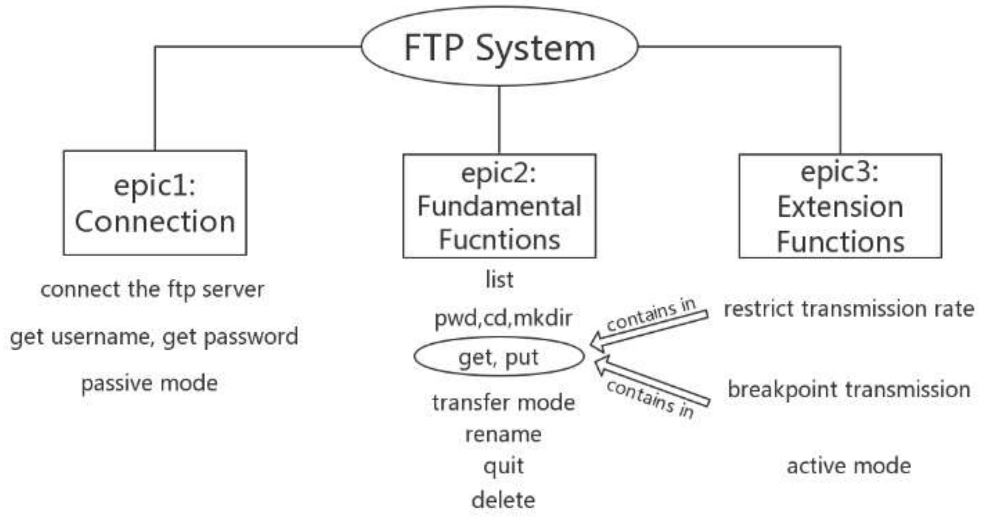
This project aims to help students to deeply understand how two hosts to establish FTP connection and how FTP constructions operate, which is based on Linux command line terminal. It requires us to design a user friendly FTP operation platform, on which user can connect FTP server using TCP protocol, receive FTP server replies in natural language, login with username and hidden password, and send commands in natural language, which includes ls, pwd, cd, mkdir, put, get, delete, rename, transfer mode and quit, etc. Moreover, the platform needs to handle errors to avoid user's improper operation or network problems according to the return number.

2. Requirements Analysis

2.1 Development Environment

Development environment:linux operating system, virtual Box-5.2.26.
Programming language: C Socket programming.

2.2 Functional requirements in details



To properly design the system, we decompose the whole tasks into three epics, then further decompose each epic into detailed functions to realize.

The first epic is connection, which includes connecting the ftp server, getting username, getting password, passive mode and sending command.

Next is fundamental functions, which includes list, pwd, cd, mkdir, uploading and getting a file, deleting a file, renaming a file, transferring mode and exit.

The final one is extension requirements, respectively are active mode(refers to connection), resuming transmission from break-point(part of get and put) and limiting transmission data rate(part of get and put).

However, a user friendly system needs not only the base and extension functions, but also error warnings when something wrong happens. So we also take error controls into consideration by handle the return number for each time command returns.

3. Preliminary Design

3.1 Decomposition of functional modules

3.1.1 Connect to server -> cliopen()

(1) Cliopen()

Connect to ftp server.

It should support both the domain name and ip address

(2) Recv()

Get return number from ftp. 220 indicates connect success

3.1.2 Log in -> login_server()

(1) Getpassword() GetUserName()

Get username and password from user.

(2) SendCommandtoFtp()

Send "USER+user input" and "PASS+user input" to ftp, and get return value from ftp server.

(3) Validate the return value

Follow the return number to do the error handling.

3.1.3 Main loop->main()

(1) Scanf()

Get user's command, then jump into detailed function modules accordingly.

3.1.4 Function modules that only involve control socket

(Pwd->ftp_pwd(), cd->ml_cwd(), mkdir->ml_mkd(), del->ml_deletefile(), ctype->ml_type(), quit->quit_toFtp(), rename->ml_renamefile())

(1) SendCommandtoFtp()

Send Command+parameter to ftp server, and get return value from ftp server.

(2) Validate the return value

Follow the return number to do the error handling.

3.1.5 Function modules that both involve control socket+data socket

(ls->ml_list(), put->ml_put(), get->ml_get())

(1) **If the data connection is passive mode->PasvMode()**

a. SendCommandtoFtp()

Send PASV to ftp server, and get return value from ftp server.

b. Retrive the IP and port specified by server

Retrive the ip address and port for data connection from the return value.

c. Cliopen()

Connect to the ip address and port for data connection. This would trigger a data socket

(2) If the data connection is active mode->ActiveMode()

a. Create a new socket to listen all the connection that comes into 13272 port.

b. Bind()

Bind the socket

c. Listen()

Start listening 13272 port

d. SendCommandtoFtp()

Send PORT+IP+Port, to tell the ftp server that I'm waiting in IP+Port, please send data to this address, and get return value.

e. Validate the return value

Follow the return number to do the error handling.

f. SendCommandtoFtp()

Send CMD+Parameter to ftp server, and get return value.

g. Validate the return value

Follow the return number to do the error handling.

h. Accept()

Accept the data connection from ftp server. This would trigger a data socket.

(3) SendCommandtoFtp()

Send CMD+Parameter to ftp server

(4) Recv()

Then we should receive message from data socket and stored it in buffer.

a. For ls: print the buffer on the screen.

b. For get: Open the file in our virtual machine. Then write the received message to the file

c. For put: Read the file content into buffer, then send to ftp server.

3.1.6 **Breakpoint downloading->ml_bpGet()**

(1) Follow the "get" command as described above.

(2) The transmission has been interrupt.

(3) ml_bpGet()

Download again, now we should retrieve the current length of the file.

(4) ml_type()

The transfer mode should be converted to binary mode.

(5) SendCommandtoFtp()

Send REST+Current length to ftp server, and get return value.

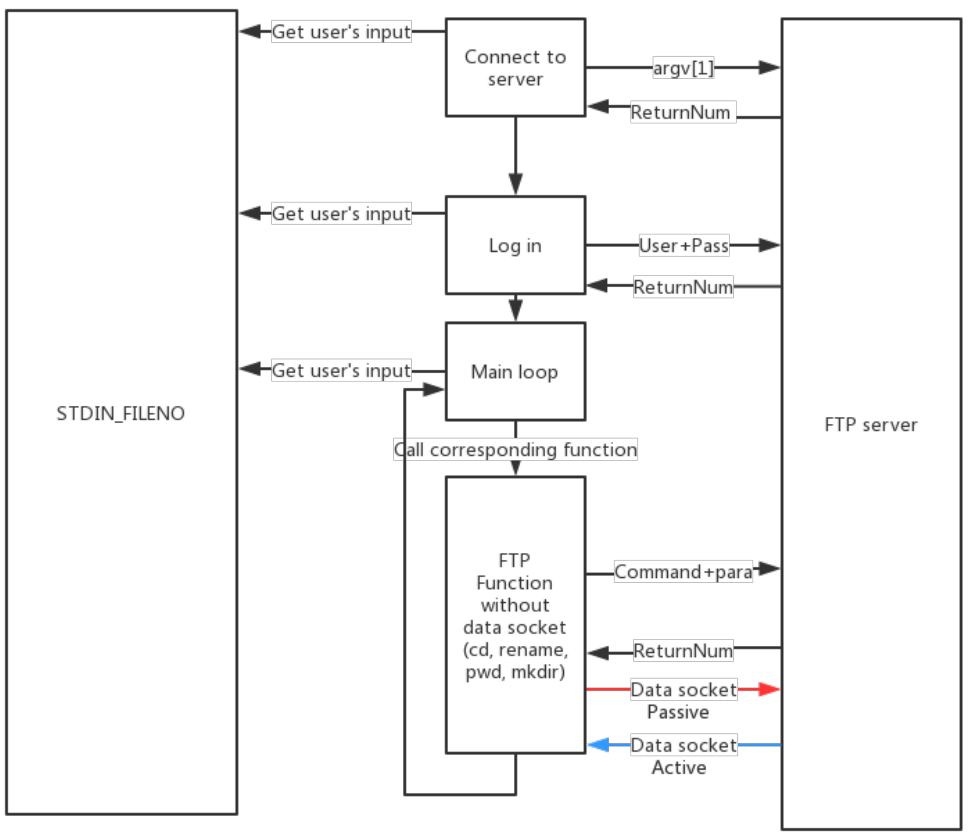
(6) Validate the return value

Follow the return number to do the error handling.

(7) Open() recv() write()

Open the file in our virtual machine. Then write the received message to the file

3.2 Relationship between modules and overall flow chart



3.3 Design of data structures

Int returnNum: hold return value (only int value) from ftp server
Char* return_bufArr[520]: hold return string, including describing fields from ftp server.
Int speed: record the speed specified by user.
Int bp: record the breakpoint last time
Char* type=A: the deault transfer type is ASCII.
Char* ftphostName: Record the the ftp server name or ip address
Int sockC: Control socket
Int d_socket: Data socket
Int sockLsnNum: Listening socket
Char* cmd: hold the command that would be sent to the ftp server
Char* bufArr: receive the message from the ftp server
Char* command: get user command and parameter from standard input.
Int handle: open file in ftp server or virtual machine.

4. Detailed Design

4.1 Epic1: about connection->cliopen() login_server()

(1)Cliopen()

Connect to ftp server.

It should support both the domain name and ip address

(2)Recv()

Get return number from ftp. 220 indicates connect success

(3)Getpassword() GetUserName()

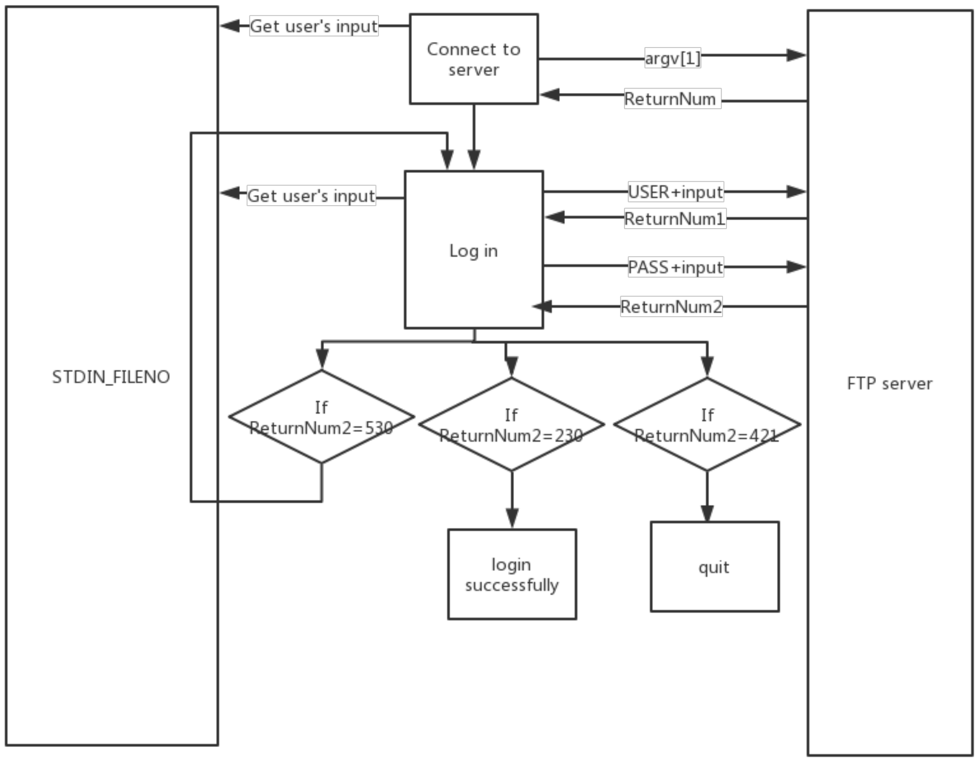
Get username and password from user.

(4)SendCommandtoFtp()

Send “USER+user input” and “PASS+user input” to ftp, and get return value from ftp server. We use *int returnNum* to hold return value (only int value) from ftp server, and *Char* return_bufArr[520]* to hold return string that includes describing fields from ftp server.

(5) Validate the return value

Follow the return number to do the error handling. If the return number is 530, it means that the password is not correct, then we should get into a loop until the return number is 230. If the return number is 421, which means the user has entered the wrong password for 3 times, then the connection would be terminated.



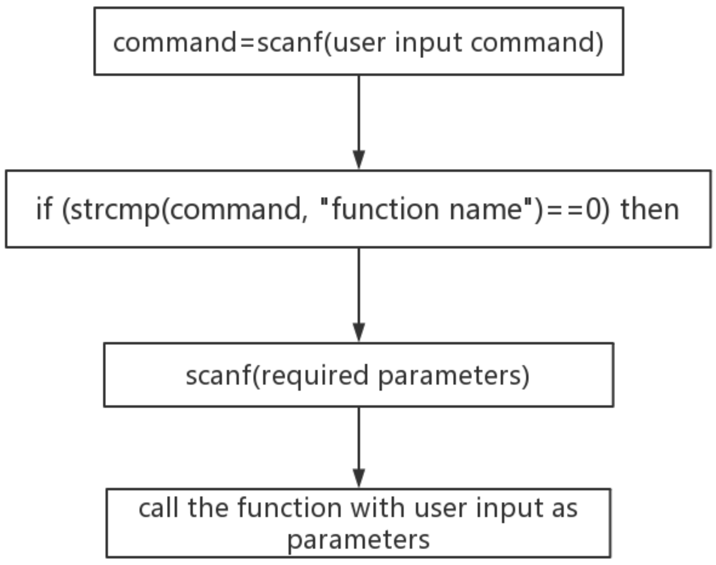
4.2 Epic2: fundamental requirements

4.2.1. Global attributes

return_buffer: to hold return string from FTP server. After each time we use it, we need to clear it out(This step is applied in sendCommandToftp).
type: to hold transfer mode
speed: to hold transmission speed

4.2.2 Main design

By using scanf, main function in our design is mainly used to absorb user input and then put the user input as parameters into the calling functions. Before doing so, it needs to figure out what command the user is going to use by using strcmp() to compare the user input with specific function calling string. Its operating logic is as follows.


command=scanf(user input command)
if (strcmp(command, "function name")==0) then
scanf(required parameters)
call the function with user input as parameters

4.2.3 Function modules that only involve control socket

Key attributes used in each module

buffArr: local buffer to hold commands, after each time we use it, we need to clear it out.
re: return number to handle error

Command for each module

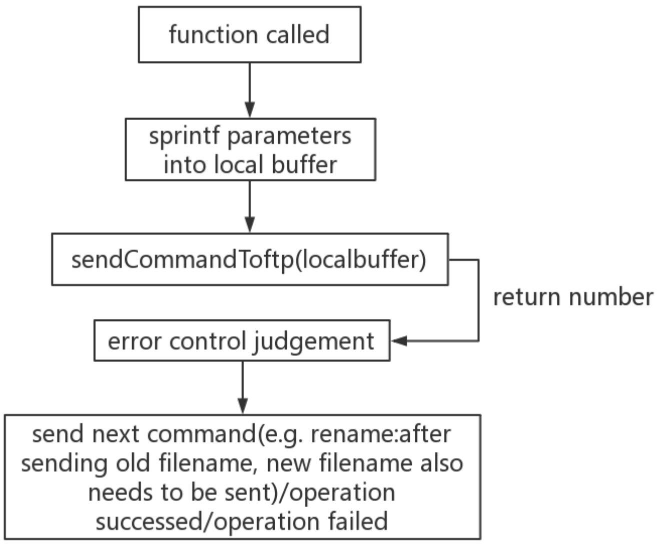
pwd: PWD
cd: CWD + path
mkdir: MKD +filename
delete: DELE + filenme
rename: RNFR + old filename; RNTO + new filename
transfer mode: TYPE A(ascii)/TYPE I(binary)
quit: QUIT

Error control for each module

pwd: if `re >= 300 || re == 0`, print the content of return buffer

cd: if re != 250, printf("The path doesn't exist.\n");
mkdir: if re == 550, printf("Directory already existed or you have no permission to make a directory.\n");
delete: if re == 550, printf("The file does not exist\n"); if re == 450, printf("You have no permission to delete.\n");
rename: if re != 350, printf("Sorry, the file does not exist.\n"); else if re != 250, print the content of return buffer
transfer mode: if re != 200, print the content of return buffer
quit: QUIT: if re >=400, print the content of return buffer

For pwd, cd, mkdir, delete, rename, transfer mode and quit, they only involves control socket, so we take them into the same consideration. The logic flow chart is showed below.


function called
sprintf parameters into local buffer
sendCommandToftp(localbuffer)
return number
error control judgement
send next command(e.g. rename:after sending old filename, new filename also needs to be sent)/operation successed/operation failed

4.2.4 Function module that also involve data socket

Key attributes

buffArr: local buffer to hold commands, after each time we use it, we need to clear it out.
re: return number to handle error

(1) Passvie mode->PasvMode()

a. SendCommandtoFtp()
   Send PASV to ftp server, and get return value.
b. Retrieve the IP and port specified by ftp server
   Retrieve the ip address and port for data connection from the return value.

c. Cliopen()

Connect to the ip address and port for data connection. This would trigger a data socket

(2) Active mode->ActiveMode()

a. Create a new socket to listen all the connection that comes into 13272 port.

b. Bind()

Bind the socket

c. Listen()

Start listening 13272 port

d. SendCommandtoFtp()

Send PORT+IP+Port, to tell the ftp server that I'm waiting in IP+Port, please send data to this address, and get return value.

e. Validate the return value

Follow the return number to do the error handling.

f. SendCommandtoFtp()

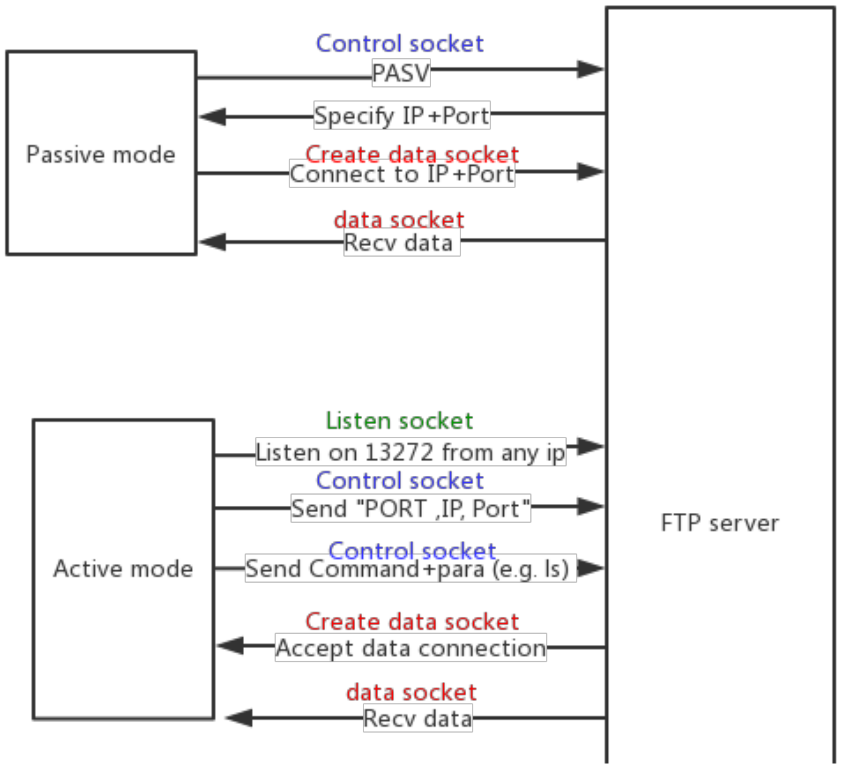
Send CMD+Parameter to ftp server, and get return value.

g. Validate the return value

Follow the return number to do the error handling.

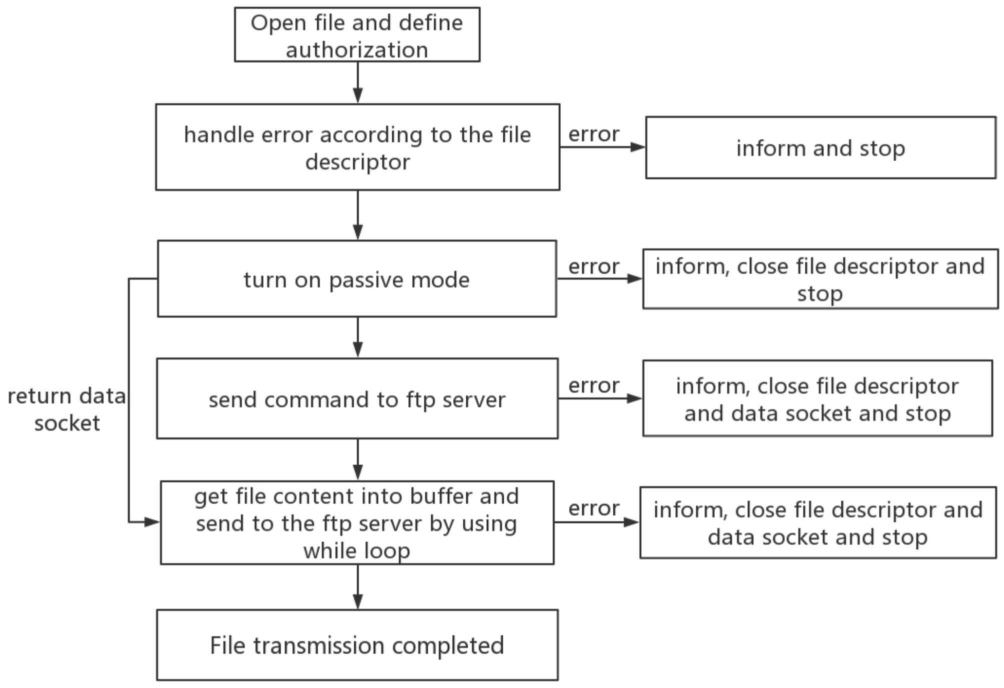
h. Accept()

Accept the data connection from ftp server. This would trigger a data socket.



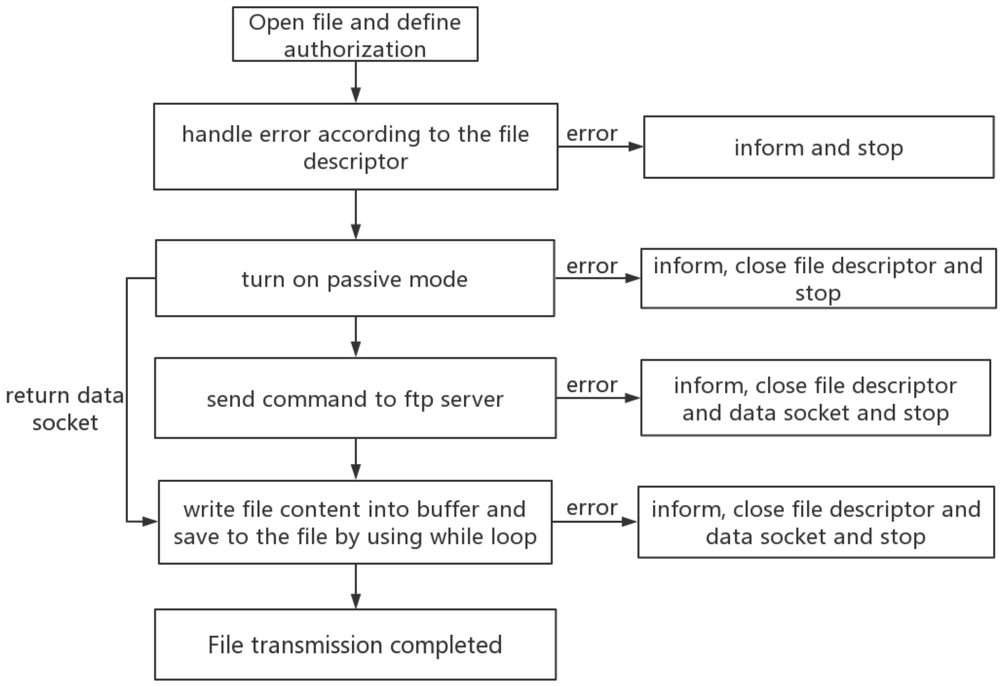
(3) Put->ml_put()

First of all, we need to open the file we need to put, defining the file operation authorization(O_RDONLY) and get the file descriptor(handle error if the file does not exist). Then apply the passive mode(handle error if passive mode fails) to get the data transfer socket and send command “STOR filename” to the ftp server(handle error according to return number. If re == 553, it means there’s a file with the same name on the ftp server. Else if re >300) or re==0, it means the user has no permission to put the file). Next, we comes to read the file content into buffer and send to the ftp server by using a while loop(handle error if sent length does not equal to read length by closing the file descriptor and data socket, and inform user about it.). If there’s no error happens, the file is successfully put, then close the file descriptor and data socket.


Open file and define authorization
handle error according to the file descriptor
error
inform and stop
turn on passive mode
error
inform, close file descriptor and stop
return data socket
send command to ftp server
error
inform, close file descriptor and data socket and stop
get file content into buffer and send to the ftp server by using while loop
error
inform, close file descriptor and data socket and stop
File transmission completed

(4) Get->ml_get()

First of all, we need to open the file we need to put, defining the file operation (O_RDWR|O_CREAT|O_TRUNC)and user authorization(S_IREAD|S_IWRITE). So we get the file descriptor(handle error if the file does not exist). Then apply the passive mode(handle error if passive mode fails) to get the data transfer socket and send command “RETR filename” to the ftp server(handle error according to return number. If re >300 or re==0, it means the file does not exist). Next, we comes to write the file content into buffer then save the data from buffer into file by using a while loop(handle error if receive length does not equal to write length by closing the file descriptor and data socket, and inform user about it.). If there’s no error happens, the file is successfully got, then close the file descriptor and data socket.


Open file and define authorization
handle error according to the file descriptor
error
inform and stop
turn on passive mode
error
inform, close file descriptor and stop
return data socket
send command to ftp server
error
inform, close file descriptor and data socket and stop
write file content into buffer and save to the file by using while loop
error
inform, close file descriptor and data socket and stop
File transmission completed

(5) List file->ml_list()

a. Enter passive mode or active mode

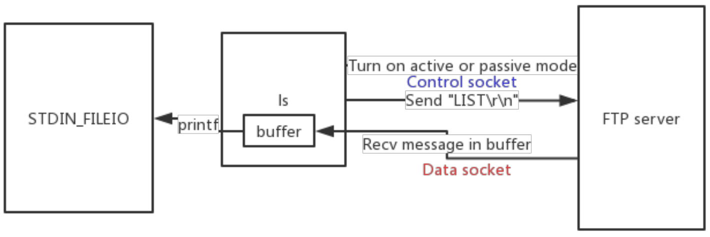
b. SendCommandtoFtp()

Send "LIST \r\n" to ftp server

d. Recv()

Then we should receive message from data socket and stored it in buffer.

e. print the buffer on the screen.

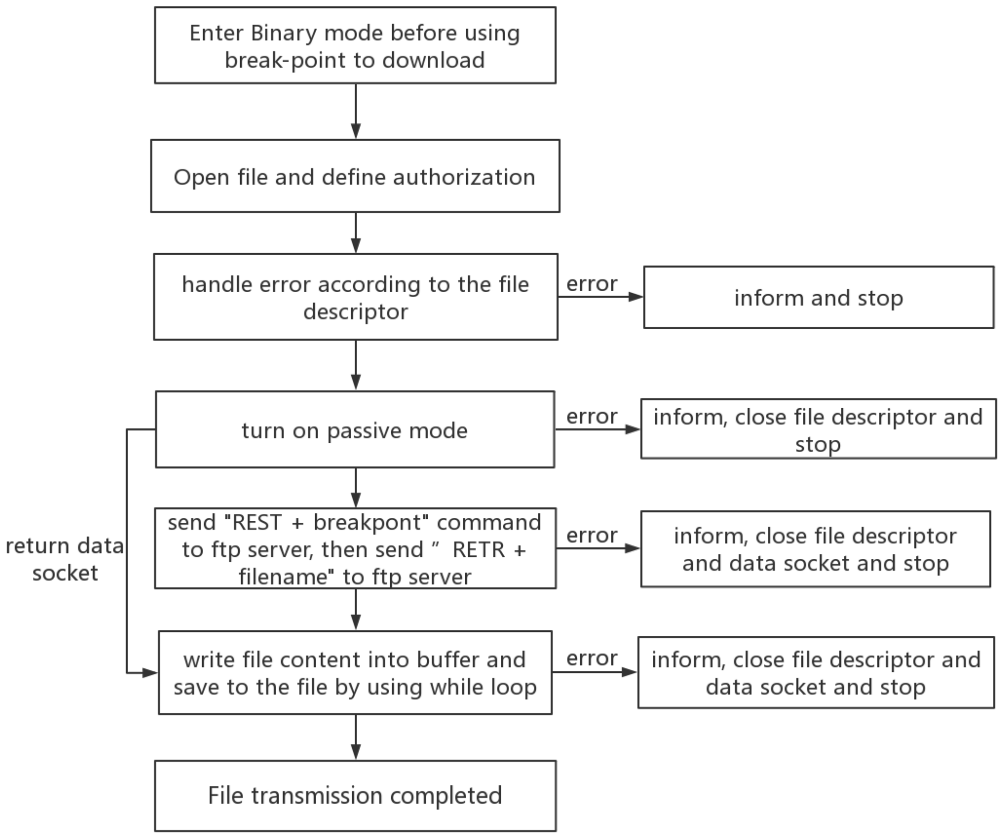


4.3 Epic3: Extension

4.3.1 Resuming download from break-point->ml_bpGet()

We assume that this module is applied when there is abnormal exit happened on client side to stop the whole system. When this situation happens, the user can first apply "ls" command to get the current length of the received file. Then take the current length as break-point to call the ml_bpGet() function. So that the user can download the file from the break-point. The internal function calling logic and logic flow chart is showed below.

a. call ml_type() to get into binary mode
b. open()
use O_APPEND instead of O_TRUNC
c. Enter passive mode
d. SendCommandtoFtp()
    Send “REST break-point\r\n” to ftp server
e. SendCommandtoFtp()
    Send”RETR filename\r\n”
f. While loop recv() + write()
g. close()


Enter Binary mode before using break-point to download
Open file and define authorization
handle error according to the file descriptor
error
inform and stop
turn on passive mode
error
inform, close file descriptor and stop
send "REST + breakpont" command to ftp server, then send " RETR + filename" to ftp server
error
inform, close file descriptor and data socket and stop
return data socket
write file content into buffer and save to the file by using while loop
error
inform, close file descriptor and data socket and stop
File transmission completed

4.3.2 Active mode detail realization has been showed at 4.2.4

4.3.3 Restrict Transmission Rate

We realize this function by firstly define the speed for recv() in each while loop(speed= len/timeuse). Timeuse is the difference between loop starttime and loop endtime. Then we apply the judgement, if the speed exceeds 5.0, then we will call upsleep() to hang up the whole process for a delay time(delay = len/5.0-timeuse), so as to reduce the speed to nearly zero. We test the transmission rate restriction in “get”. The internal function calling logic and logic

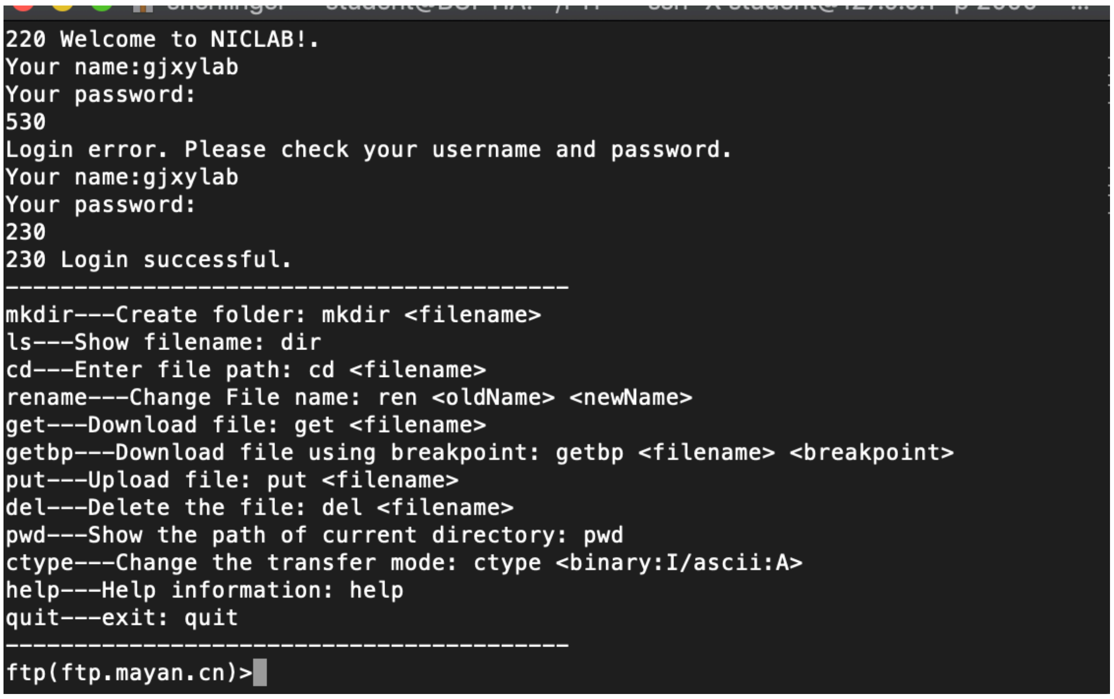
flow chart is showed below.
Variation only into the while loop:

a. getstarttime()
b. recv()
c. write()
d. getendtime()
e. Calculate timeuse and speed
f. Claculate delay
g. If speed>5.0
h. Call upsleep(delay)

5 Results

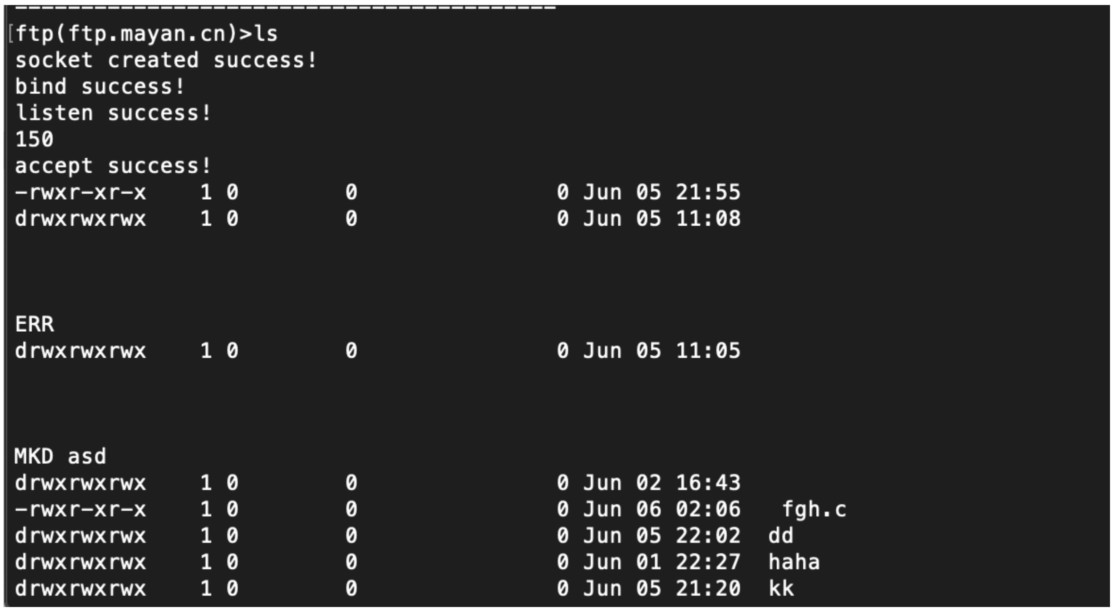
(1) Log in

if you input the wrong password, our application will prompt you to enter again.



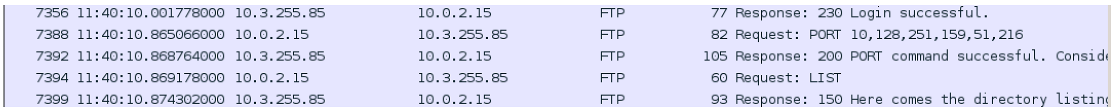
```
220 Welcome to NICLAB!.
Your name:gjxylab
Your password:
530
Login error. Please check your username and password.
Your name:gjxylab
Your password:
230
230 Login successful.
-----------------------------------------
mkdir---Create folder: mkdir <filename>
ls---Show filename: dir
cd---Enter file path: cd <filename>
rename---Change File name: ren <oldName> <newName>
get---Download file: get <filename>
getbp---Download file using breakpoint: getbp <filename> <breakpoint>
put---Upload file: put <filename>
del---Delete the file: del <filename>
pwd---Show the path of current directory: pwd
ctype---Change the transfer mode: ctype <binary:I/ascii:A>
help---Help information: help
quit---exit: quit
-----------------------------------------
ftp(ftp.mayan.cn)>
```

(2) ls in active mode



```
[ftp(ftp.mayan.cn)>ls
socket created success!
bind success!
listen success!
150
accept success!
-rwxr-xr-x    1 0        0               0 Jun 05 21:55
drwxrwxrwx    1 0        0               0 Jun 05 11:08



ERR
drwxrwxrwx    1 0        0               0 Jun 05 11:05



MKD asd
drwxrwxrwx    1 0        0               0 Jun 02 16:43
-rwxr-xr-x    1 0        0               0 Jun 06 02:06   fgh.c
drwxrwxrwx    1 0        0               0 Jun 05 22:02  dd
drwxrwxrwx    1 0        0               0 Jun 01 22:27  haha
drwxrwxrwx    1 0        0               0 Jun 05 21:20  kk
```

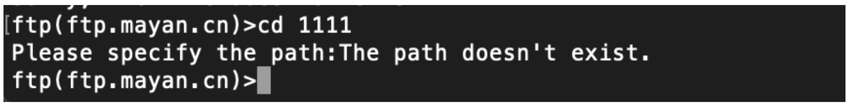
Check active mode



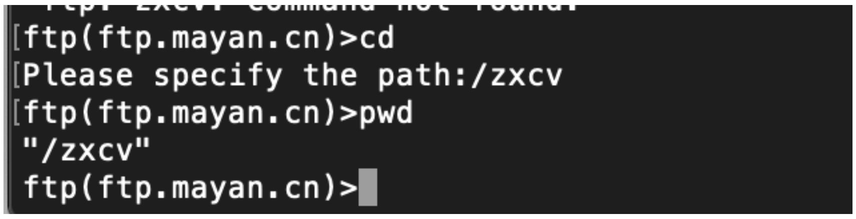
```
7356 11:40:10.001778000 10.3.255.85      10.0.2.15        FTP     77 Response: 230 Login successful.
7388 11:40:10.865066000 10.0.2.15        10.3.255.85      FTP     82 Request: PORT 10,128,251,159,51,216
7392 11:40:10.868764000 10.3.255.85      10.0.2.15        FTP    105 Response: 200 PORT command successful. Conside
7394 11:40:10.869178000 10.0.2.15        10.3.255.85      FTP     60 Request: LIST
7399 11:40:10.874302000 10.3.255.85      10.0.2.15        FTP     93 Response: 150 Here comes the directory listin
```

(3) cd

if the directory you chose doesn't exist, our application will prompt you to enter again.



```
[ftp(ftp.mayan.cn)>cd 1111
Please specify the path:The path doesn't exist.
ftp(ftp.mayan.cn)>
```



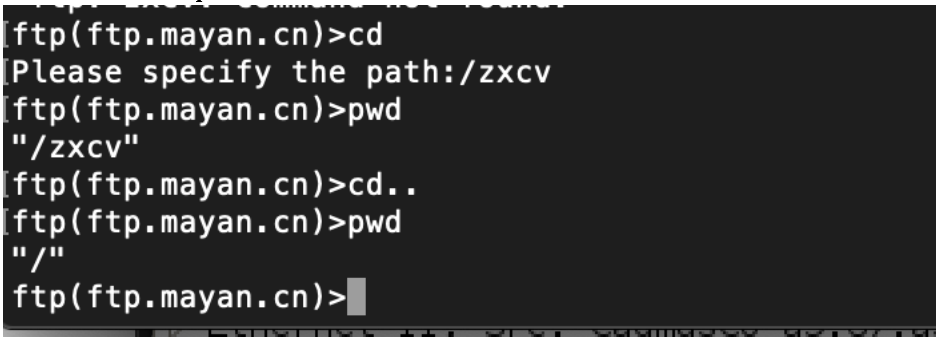
```
[ftp(ftp.mayan.cn)>cd
[Please specify the path:/zxcv
[ftp(ftp.mayan.cn)>pwd
"/zxcv"
ftp(ftp.mayan.cn)>
```

(4) cd..

You can return back to the upper directory

(5) pwd

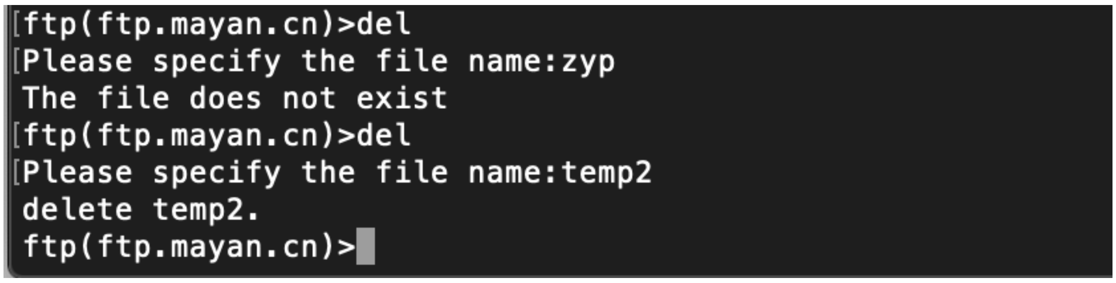
Show current path.



```
[ftp(ftp.mayan.cn)>cd
[Please specify the path:/zxcv
[ftp(ftp.mayan.cn)>pwd
"/zxcv"
[ftp(ftp.mayan.cn)>cd..
[ftp(ftp.mayan.cn)>pwd
"/"
ftp(ftp.mayan.cn)>
```

(6) del

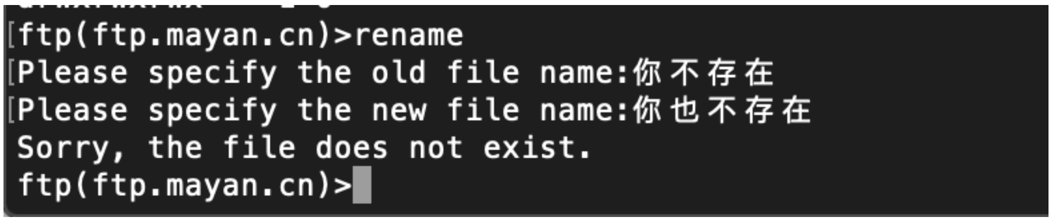
if the file you chose doesn't exist, our application will prompt you to enter again.



```
[ftp(ftp.mayan.cn)>del
[Please specify the file name:zyp
 The file does not exist
[ftp(ftp.mayan.cn)>del
[Please specify the file name:temp2
 delete temp2.
 ftp(ftp.mayan.cn)>
```

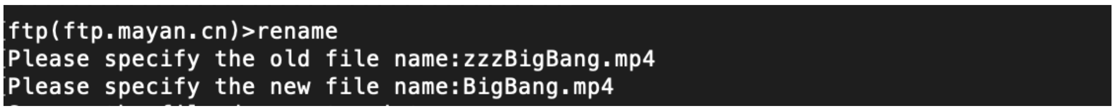
(7) rename

You can rename a file. if the file you chose doesn't exist, our application will prompt you to enter again.



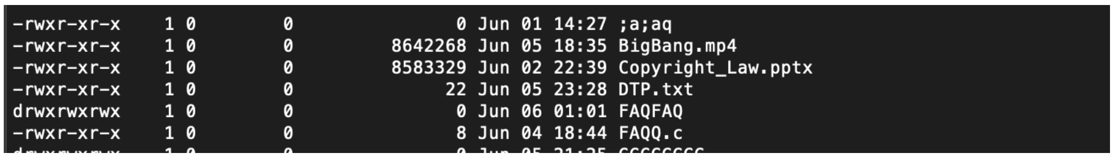
```
[ftp(ftp.mayan.cn)>rename
[Please specify the old file name:你不存在
[Please specify the new file name:你也不存在
 Sorry, the file does not exist.
 ftp(ftp.mayan.cn)>
```

Rename successfully



```
[ftp(ftp.mayan.cn)>rename
[Please specify the old file name:zzzBigBang.mp4
[Please specify the new file name:BigBang.mp4
```

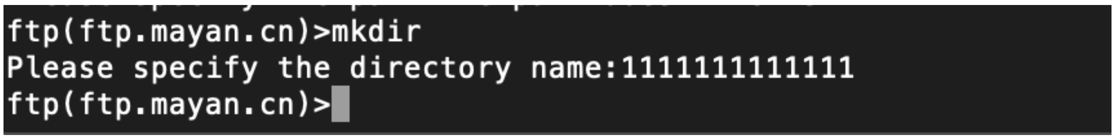
Check if it has been renamed in ftp server



```
-rwxr-xr-x    1 0        0               0 Jun 01 14:27 ;a;aq
-rwxr-xr-x    1 0        0         8642268 Jun 05 18:35 BigBang.mp4
-rwxr-xr-x    1 0        0         8583329 Jun 02 22:39 Copyright_Law.pptx
-rwxr-xr-x    1 0        0              22 Jun 05 23:28 DTP.txt
drwxrwxrwx    1 0        0               0 Jun 06 01:01 FAQFAQ
-rwxr-xr-x    1 0        0               8 Jun 04 18:44 FAQQ.c
```

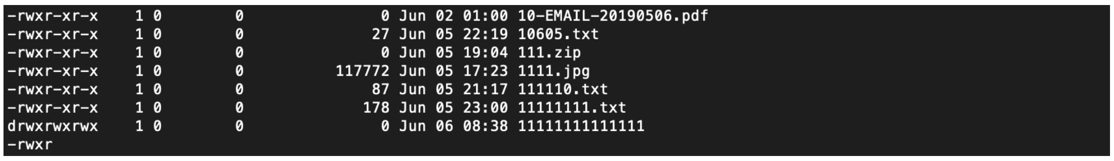
(8) mkdir

You can make a new directory.



```
ftp(ftp.mayan.cn)>mkdir
Please specify the directory name:1111111111111
ftp(ftp.mayan.cn)>
```

Check if the new directory has been created



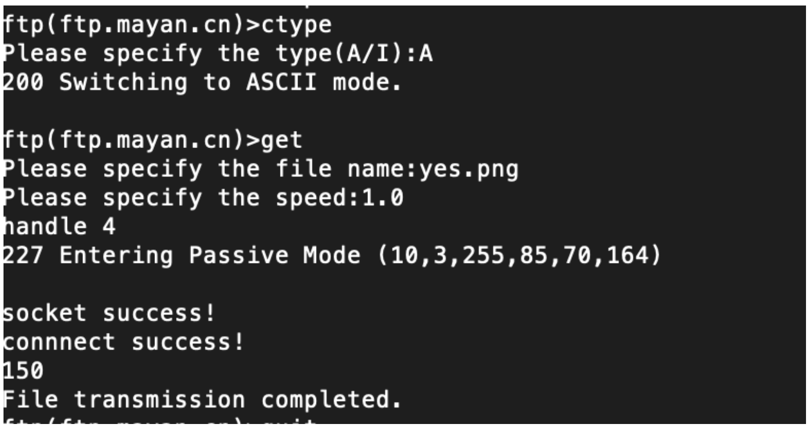
```
-rwxr-xr-x    1 0        0               0 Jun 02 01:00 10-EMAIL-20190506.pdf
-rwxr-xr-x    1 0        0              27 Jun 05 22:19 10605.txt
-rwxr-xr-x    1 0        0               0 Jun 05 19:04 111.zip
-rwxr-xr-x    1 0        0          117772 Jun 05 17:23 1111.jpg
-rwxr-xr-x    1 0        0              87 Jun 05 21:17 111110.txt
-rwxr-xr-x    1 0        0             178 Jun 05 23:00 11111111.txt
drwxrwxrwx    1 0        0               0 Jun 06 08:38 1111111111111
-rwxr
```

(9) ctype

You could change your transmission type, i.e. ASCII or Binary.

The default transfer mode is ASCII.

ASCII only supports text file, which is encoded by ASCII, while Binary will keep the file encoding scheme as it is.

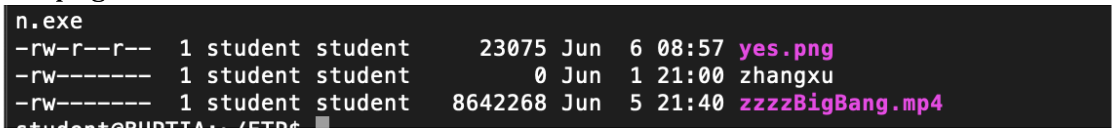


```
ftp(ftp.mayan.cn)>ctype
Please specify the type(A/I):A
200 Switching to ASCII mode.

ftp(ftp.mayan.cn)>get
Please specify the file name:yes.png
Please specify the speed:1.0
handle 4
227 Entering Passive Mode (10,3,255,85,70,164)

socket success!
connnect success!
150
File transmission completed.
```

(10) put (Passive mode)

You could upload your file into ftp server and specify your desired speed.

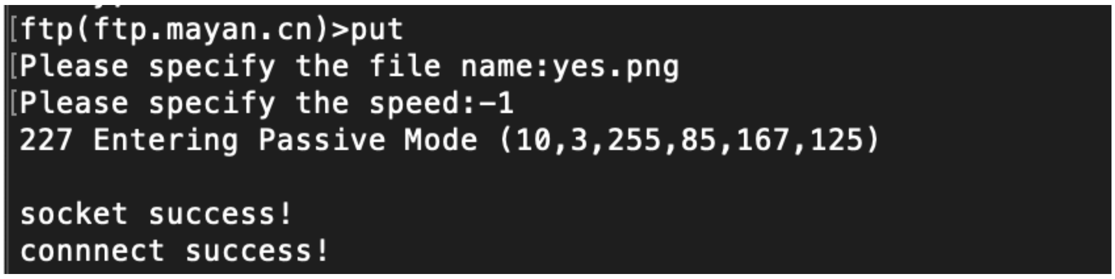
Yes.png in our virtual machine.



```
n.exe
-rw-r--r--  1 student student     23075 Jun  6 08:57 yes.png
-rw-------  1 student student         0 Jun  1 21:00 zhangxu
-rw-------  1 student student   8642268 Jun  5 21:40 zzzzBigBang.mp4
```

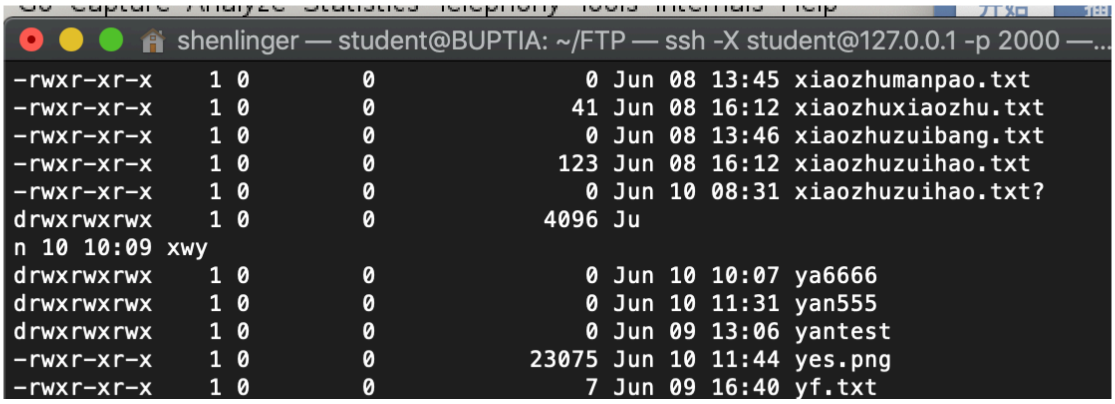
Upload to ftp server

-1 indicate original speed without restriction.



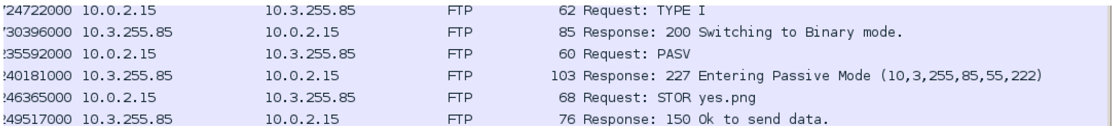
```
[ftp(ftp.mayan.cn)>put
[Please specify the file name:yes.png
[Please specify the speed:-1
 227 Entering Passive Mode (10,3,255,85,167,125)

 socket success!
 connnect success!
```

The file has been uploaded to ftp server, and the file length is the same as the one in virtual machine.



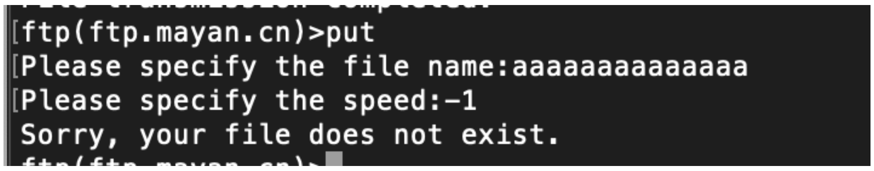
```
shenlinger — student@BUPTIA: ~/FTP — ssh -X student@127.0.0.1 -p 2000 —...
-rwxr-xr-x    1 0        0               0 Jun 08 13:45 xiaozhumanpao.txt
-rwxr-xr-x    1 0        0              41 Jun 08 16:12 xiaozhuxiaozhu.txt
-rwxr-xr-x    1 0        0               0 Jun 08 13:46 xiaozhuzuibang.txt
-rwxr-xr-x    1 0        0             123 Jun 08 16:12 xiaozhuzuihao.txt
-rwxr-xr-x    1 0        0               0 Jun 10 08:31 xiaozhuzuihao.txt?
drwxrwxrwx    1 0        0            4096 Ju
n 10 10:09 xwy
drwxrwxrwx    1 0        0               0 Jun 10 10:07 ya6666
drwxrwxrwx    1 0        0               0 Jun 10 11:31 yan555
drwxrwxrwx    1 0        0               0 Jun 09 13:06 yantest
-rwxr-xr-x    1 0        0           23075 Jun 10 11:44 yes.png
-rwxr-xr-x    1 0        0               7 Jun 09 16:40 yf.txt
```

Check Passive Mode



```
'24722000  10.0.2.15      10.3.255.85    FTP    62 Request: TYPE I
'30396000  10.3.255.85    10.0.2.15      FTP    85 Response: 200 Switching to Binary mode.
35592000   10.0.2.15      10.3.255.85    FTP    60 Request: PASV
40181000   10.3.255.85    10.0.2.15      FTP   103 Response: 227 Entering Passive Mode (10,3,255,85,55,222)
46365000   10.0.2.15      10.3.255.85    FTP    68 Request: STOR yes.png
49517000   10.3.255.85    10.0.2.15      FTP    76 Response: 150 Ok to send data.
```

If the file name doesn't exist in your virtual machine, our application will prompt you to enter again.

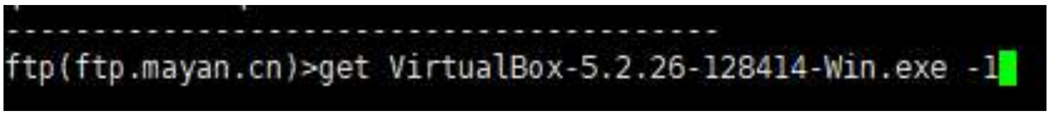


```
ftp(ftp.mayan.cn)>put
Please specify the file name:aaaaaaaaaaaaaa
Please specify the speed:-1
Sorry, your file does not exist.
```

(11) get (Passive mode)

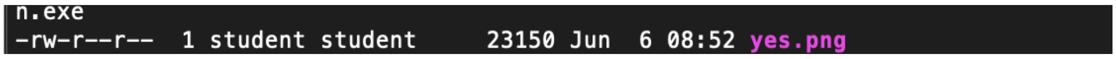
Delete the Yes.png in our virtual machine, download it from ftp server again and specify your desired speed.

-1 indicate original speed without restriction.

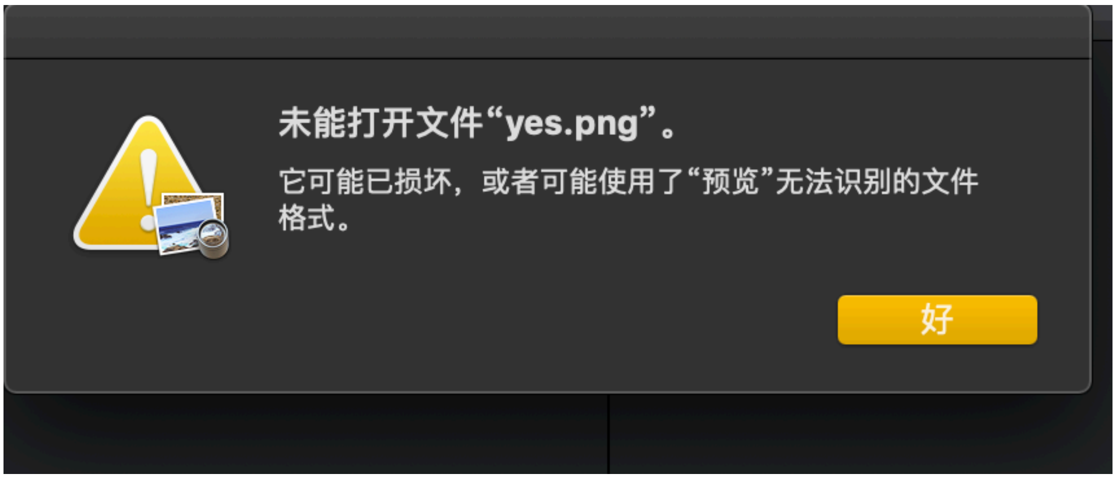


```
-----------------------------------------
ftp(ftp.mayan.cn)>get VirtualBox-5.2.26-128414-Win.exe -1
```

While the transfer mode is ASCII, we found that the end file could not be opened as a png file. And the file length is not the same as the original file.



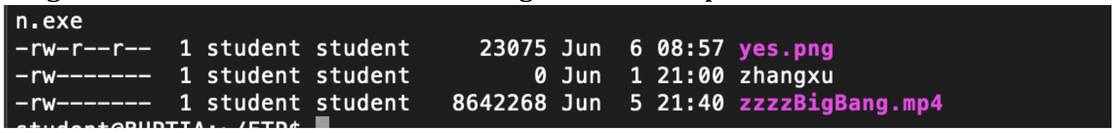
```
n.exe
-rw-r--r--  1 student student     23150 Jun  6 08:52 yes.png
```



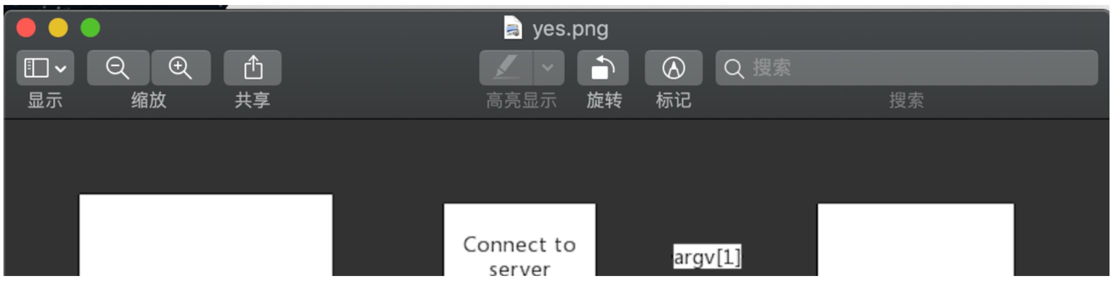
未能打开文件“yes.png”。

它可能已损坏，或者可能使用了“预览”无法识别的文件格式。

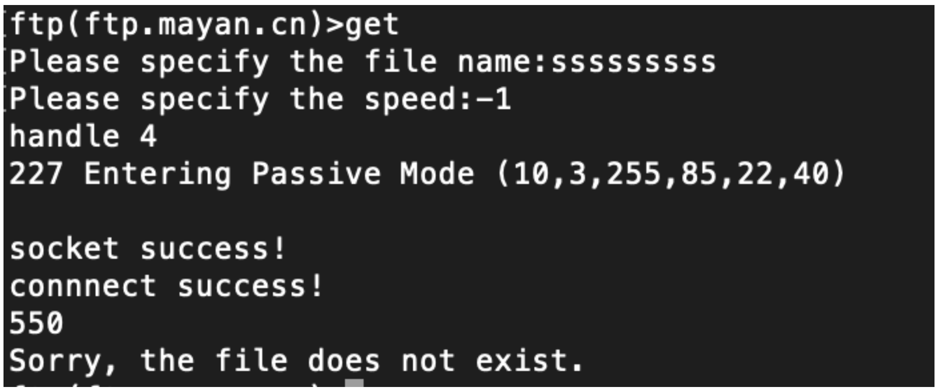
And while the transfer mode is Binary, the end file could be opened. And now the length of the file is the same as the original one in ftp server.



```
n.exe
-rw-r--r--  1 student student     23075 Jun  6 08:57 yes.png
-rw-------  1 student student         0 Jun  1 21:00 zhangxu
-rw-------  1 student student   8642268 Jun  5 21:40 zzzzBigBang.mp4
```



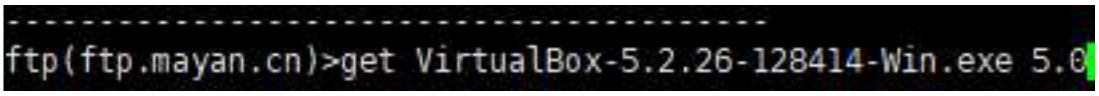
if the file you chose doesn't exist, our application will prompt you to enter again.



```
ftp(ftp.mayan.cn)>get
Please specify the file name:sssssssss
Please specify the speed:-1
handle 4
227 Entering Passive Mode (10,3,255,85,22,40)

socket success!
connnect success!
550
Sorry, the file does not exist.
```

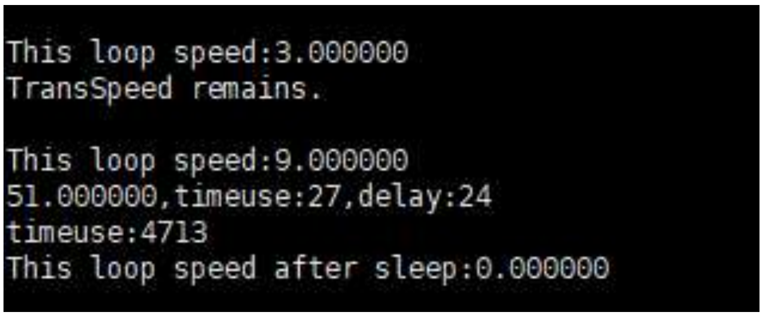
(12) restrict transmission rate

Here, you can indicate your desire speed in kbps.



```
----------------------------------------
ftp(ftp.mayan.cn)>get VirtualBox-5.2.26-128414-Win.exe 5.0
```

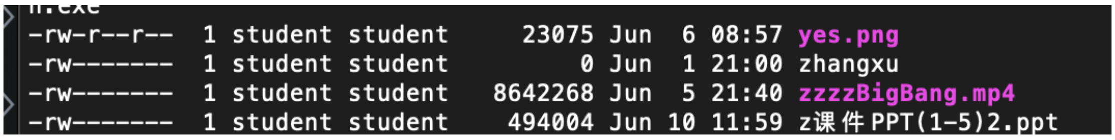
Our application will restrict your average speed to less than 5 kbps.



```
This loop speed:3.000000
TransSpeed remains.

This loop speed:9.000000
51.000000,timeuse:27,delay:24
timeuse:4713
This loop speed after sleep:0.000000
```

(13) getbp

If your downloading is interrupted, you could specify a breakpoint next time using getbp, and proceed your transmission from the breakpoint.

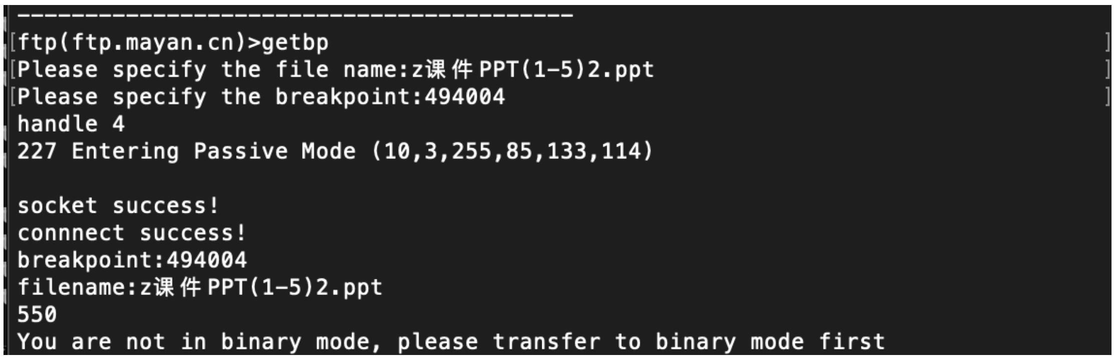


```
 n.exe
 -rw-r--r--  1 student student     23075 Jun  6 08:57 yes.png
 -rw-------  1 student student         0 Jun  1 21:00 zhangxu
 -rw-------  1 student student   8642268 Jun  5 21:40 zzzzBigBang.mp4
 -rw-------  1 student student    494004 Jun 10 11:59 z课件PPT(1-5)2.ppt
```

Before breakpoint transmission, the “z 课件 PPT(1-5)2” file has transmitted 494004/1549812 byte (transmitted/total).

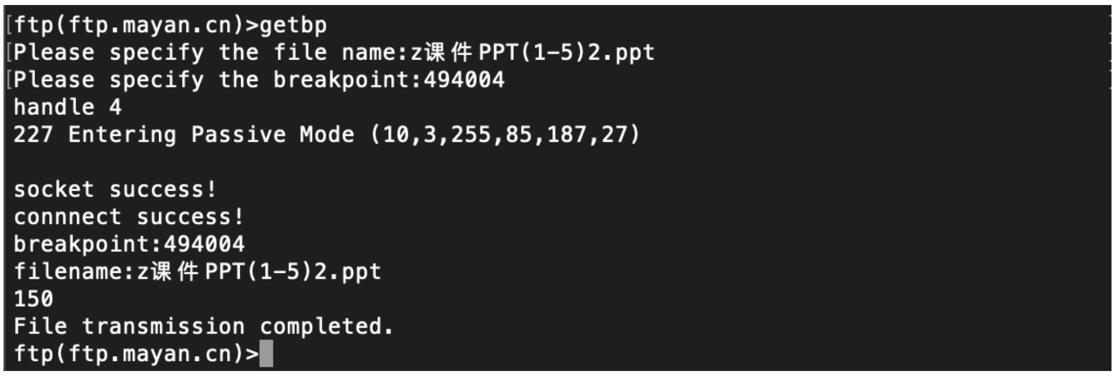
So we use getbp to set the breakpoint and proceed our transmission from 494004.

If transfer mode is not binary, we would prompt the convert the transfer mode, since the breakpoint transmission only supports binary mode.



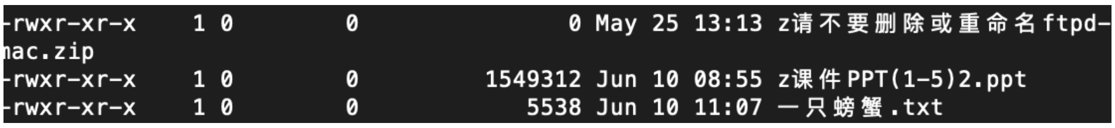
```
-----------------------------------------
ftp(ftp.mayan.cn)>getbp
Please specify the file name:z课件PPT(1-5)2.ppt
Please specify the breakpoint:494004
handle 4
227 Entering Passive Mode (10,3,255,85,133,114)

socket success!
connnect success!
breakpoint:494004
filename:z课件PPT(1-5)2.ppt
550
You are not in binary mode, please transfer to binary mode first
```

After converting to binary mode, we could proceed our transmission.



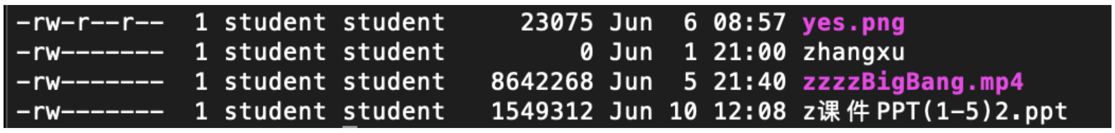
```
ftp(ftp.mayan.cn)>getbp
Please specify the file name:z课件PPT(1-5)2.ppt
Please specify the breakpoint:494004
handle 4
227 Entering Passive Mode (10,3,255,85,187,27)

socket success!
connnect success!
breakpoint:494004
filename:z课件PPT(1-5)2.ppt
150
File transmission completed.
ftp(ftp.mayan.cn)>
```

Compare the file length with the original one in ftp server.
"z 课件…" in Ftp server



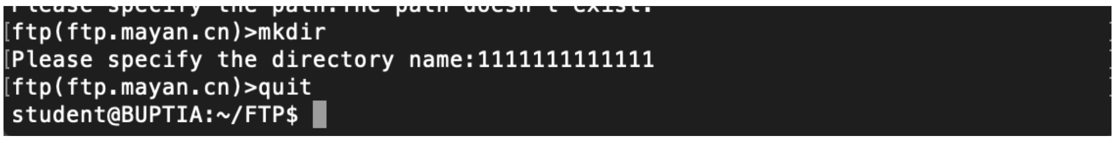
```
-rwxr-xr-x    1 0        0              0 May 25 13:13 z请不要删除或重命名ftpd-
mac.zip
-rwxr-xr-x    1 0        0        1549312 Jun 10 08:55 z课件PPT(1-5)2.ppt
-rwxr-xr-x    1 0        0           5538 Jun 10 11:07 一只螃蟹.txt
```

"z 课件…" in virtual machine



```
-rw-r--r--  1 student student     23075 Jun  6 08:57 yes.png
-rw-------  1 student student         0 Jun  1 21:00 zhangxu
-rw-------  1 student student   8642268 Jun  5 21:40 zzzzBigBang.mp4
-rw-------  1 student student   1549312 Jun 10 12:08 z课件PPT(1-5)2.ppt
```

(14)   quit



```
Please specify the path:The path doesn't exist!
ftp(ftp.mayan.cn)>mkdir
Please specify the directory name:1111111111111
ftp(ftp.mayan.cn)>quit
student@BUPTIA:~/FTP$
```

5. Summary and Conclusion

5.1 Individual contribution and self evaluation:

(1) Jincheng LIU

Responsible for commands, including get, put, mkdir, cd, rename, ctype, delete. Also responsible for breakpoint transmission and restricting transmission rate.

Self Evaluation:

(2) Linger SHEN

Responsible for overall control connection between ftp server and user, including getting user's command, connecting, sending command to ftp server, receiving and analyzing response from it. Also responsible for data connection, including active mode and passive mode, plus "ls" command.

Self Evaluation:

From this ftp project, we write the codes ourselves and learn a lot from it. I'm more familiar with how ftp protocol works, including how control connetction and data connection interact with each other in active mode and passive mode. And I incorporate my socket programming knowledge that I learnt in the class, plus I learn some programming technique from this practive, such as scanf, sprint, select.

If I have more time to improve my code, I would handle the unexpected connection's breakup. In another word, If the server send 421 back, then we would notify the user the connection has been interrupted by timeout or bad network environment, and futhermore lead to a safe quit.

5.2 Conclusion:

(1) From this project, we combine our knowledge of C Socket programming that we learnt in the class, including TCP connection, address resolving, read and write file, TCP data transmission.

(3) And we also learn some techinique in C socket programming, e.g. sscanf, sprint, select. And we further improved out socket programming skill.

(4) We get familiar with the basic command of ftp.

(5) We figure out how active mode and passive mode works.

(6) We explore some extension functions, i.e. transmission speed restriction and breakpoint downloading. It brings us to think more about the process of ftp transmission.