Socket Programming Report

Xi Ye 叶曦
2014013417

Experiment Environment

Virtual Machine
* **OS**: ubuntu 15.10
* **memory**: 3.8G
* **Processor**: Intel Core i5-4210U CPU @ 1.70GHz X 4

Overview

Server: introduction to the server.

Client: introduction to the client.

Feature: Valuable feaure of my work.

Experince: Experiences with the project.

Server

Accepted Arguements

Run server using
`sudo ./server [-port PORT] [-root ROOT]`
If you want, you can specify the port and root for server using the optional [-port] and [-root] arguements. And the defualt setting is

- port: 21
- root: /tmp

Supported Verbs

USER & PASS: Use USER [username] and PASS [password] to log in the FTP server. USER anonymous allows guests loging in. User verification is implemented and user-table is stored in `server/src/user.config`.
PORT & PASV: PORT [IP and PORT] or PASV are used to set the mode of FTP server. If PORT is set, the client opens a socket and waits the server to connect. If PASV is set, the server sends IP and PORT to the client and waits for the client. PORT and PASV is one-time used.
RETR: RETR [file] retrieval [file] from FTP server.
STOR: STOR [file] upload [file] to FTP server.
SYST: on receiving a SYST command, return the string “215 UNIX Type: L8” to the client.
TYPE: on receiving a “TYPE I” command, return “200 Type set to I.” to the client. On receiving some other TPYE command, return an appropriate error code.
CWD: CWD [path] asks the server to set the name prefix to this pathname, or to another pathname that will have the same effect as this pathname if the filesystem does not change.
CDUP: CDUP asks the server to set the name prefix to upper directory.
DELE: DELE [file] asks server to delete the [file].
LIST: LIST [path/file] asks server to list the file or directories in target path or show information of file. If empty paremeters are passed, the server show the list of the current name prefix. Usually, there could be a number of files or directories in a directory. So, the server use a file-transfer style to response to the LIST verb. And before List, PORT or PASV should be set. **MKD:** MDK [dir] create a new directory of server.
PWD: PWD show the name prefix of the server.
RMD: RMD [dir] removes an empty directory of server. Notice that [dir] must be an empty directory.
RNFR & RNTO: RNFR [src] and RNTO [dst] should be used as a union.

Client

Accepted Arguements

Run server using

```
sudo ./server [-ip IP] [-port PORT]
```

If you want, you can specify the host ip and host port for client using the optional [-ip] and [-port] arguements. And the defualt setting is

- ip:127.0.0.1
- port: 21

Surported Commands

After sending your username and password, you enter the FTP Command Prompt, where the following commands are suported. Use **help** command to view all the supported commands. Here we only select some important commands.
help: show the supported commands.
upload & download: upload [src] [dst] upload the [src] file to the server and save the file with name [dst] (example: `upload a.dat b.dat`).
download [src] [dst] download [src] file from server and save the file with name [dst] (example: `download b.dat a.dat`)
setport & setpasv: set the transfer style to PASV & PORT.
list: list [dir] display the file lists of server.
others: please refer to the HELP command

Feature

Well Designed Software Architecture
Unlike many others putting all codes in just one file, I devide the program into many modules. The server and the client share lots of common codes.
The **ByteStream** module deals with the byte stream of telecommunication. Both the server and client use funcions in **ByteStream** to read/write from socket. With this module, I can handle socket byte stream just like stdin and stdout.
The **FtpCommon** module deals with network requests and responses. The server and the client both use the module to open a socket and listen on it or connect to the remote host with specific IP address and port.
Multi-thread Programming
To deal with multi-client requests,the server use main thread to listen on new connection. And assign a unique thread for each client. That is, the file transfer of one client won't affect other clients and won't block the server, too.
Support All Optional Verbs and User Table
The server support all the optional verbs user verification. You can easily edit the user table.
Powerful Client Supports Both Server and Local Operation
The client support many useful commands. In parical, one may need both change the name prefix of server and change the work directory of the client. For excample, with my client, one can easily change to another directory, list the content of that direcoty, and upload the file to the FTP server.

Experince

I learned a lot from this homework assignment. At first, I thought that it was just another simple program to write. However, when I was coding, I found many problems that were hard to handle. For example, you should use `htons` to transform the port style or you client can not connect to the server. It took a long time for me to solve the bug. Also, I spent long time thinking of a better architecure for this program and finally choose the one metioned above. It turned out that with well designed architecture, I can have a clear view of my codes and implement new features quickly. Another import things I learned is the usage of Standard C Libary. I spent lots of time reading the documents from CPP reference. It helped me greatly. It is proved that to implement a FTP with C needs lots of time. I speed almost two weeks with **42 commits**, **4212++** and **1712--** (From Github) to complete the assignment. I spent almost two weeks to complete the assignment. After finishing it, I have a clear understand of FTP protocal and realize that Network Programming needs both **cautiousness** and **technical ability**.