File Transfer Protocol (FTP) Client Software

Objectives

1. To learn how a computer network protocol is outlined in a Request for Comments (RFC) document by Internet Engineering Taskforce (IETF) as an Internet Standard.
2. To understand File Transfer Protocol (FTP) from RFC 959.
3. To learn and use Unix Network or Socket programming API.
4. To implement FTP Client software according to Internet Standard outlined in RFC 959.

Specifications

You have been using **File Transfer Protocol** (FTP) to get or store files from or to FTP servers. You use a FTP Client software in your host computer to communicate with an FTP server. FTP server runs a FTP Server software. In this project, you are going to implement your own FTP Client software in C programming language.

FTP is a client-server protocol to transfer files to/from the servers. An FTP client sends request message to an FTP server regarding a file transfer. FTP server interprets the request message, takes appropriate action, and sends back response message to the client. **RFC 959** describes FTP and its request and response messages in detail. You need to read and understand **RFC 959** to complete this project. Although, **RFC 959** describes many request messages, you need to implement only the followings:

- **USER**
- **PASS**
- **PWD**
- **CWD**
- **CDUP**
- **PASV**
- **NLST**
- **RETR**
- **QUIT**

Your FTP Client software must have a command-line user interface (UI) so that the users can enter commands through this interface. Your software must accept and process following commands. Some of these commands have an argument. For example, command **user** has an argument <**username**>. A command and its argument is always space separated.

- **help**
- **user <username>**

- **pass <password>**
- **pwd**
- **dir**
- **cwd <dirname>**
- **cdup**
- **get <filename>**
- **quit**

- Command '**help**' displays the list of commands supported by this software, their syntaxes, and meanings.
- Command '**user**' sends username to the FTP server for authentication. You need to support only one user with user name 'csci460' and password '460pass'.
- Command '**pass**' sends the password to the FTP server for authentication.
- Command '**pwd**' prints the current working directory of the FTP server.
- Command '**dir**' lists the contents of the current working directory of the FTP server.
- Command '**cwd**' changes the current working directory to a specified directory. If the specified directory is beyond current working directory an error is reported by the server.
- Command '**cdup**' changes the current working directory to the parent directory. If the parent directory is beyond server's base directory, an error is reported by the server.
- Command '**get**' fetches the specified file from the FTP server. If the specified file is not available an error is reported by the server.
- Command '**quit**' terminates the software.

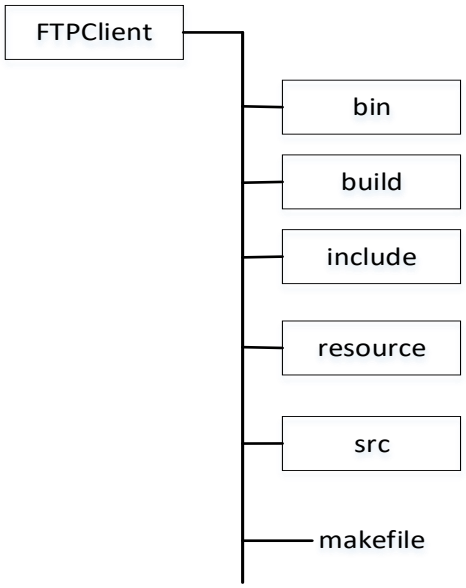
Your software must interpret above user commands, translate them into appropriate FTP request messages, send the FTP request messages to FTP server, receive the response messages from the server, and present the response to the user in a user-friendly manner.

FTP Client and Server communicate request and response over a control connection. Some request or response involve a data transmission. For example, RETR request involves a data transmission to transfer file content as the part of a successful response. FTP server uses a separate connection for each data transmission. A data connection is opened on demand and closed when a data transmission is complete. FTP server can operate either in active or in passive mode. In active mode, FTP server opens the data connection with the client on demand. If the client is behind a firewall, FTP active mode fails to open a data connection. In passive mode, when a data connection is required the server opens a connection listener so that client can send connection request to the listener and open a data connection. Client instructs the server to enter into passive mode by sending a PASV request message to the server. The server opens the connection listener on a port and sends the port number to the client in its PASV response. After receiving the PASV response, the client retrieves listener port number and sends a connection request to the listener port in order to open a data connection. In this project, you must

implement passive mode and your FTP client software, must send a PASV request before any data request, such as RETR and NLST.

Tasks

1. **Organize** your project directory as follows:



2. **Copy** this **makefile** into your **makefile**.
3. **Copy** **ftp_client.cpp** file into your **src** folder.
4. **Copy** following **hpp** files into your **include** folder.
    a. **ftp_client_command.hpp**
    b. **ftp_client_connection.hpp**
    c. **ftp_client_control.hpp**
    d. **ftp_client_ui.hpp**
    e. **ftp_server_response.hpp**
5. **Implement** following **cpp** files into your **src** folder.
    a. **ftp_client_command.cpp**
    b. **ftp_client_connection.cpp**
    c. **ftp_client_control.cpp**
    d. **ftp_client_ui.cpp**
6. The lists of internal dependencies of the **cpp** files in this project are as follows.
    i. ***ftp_client.cpp:*** *ftp_client_control.hpp, ftp_client_ui.hpp*

    ii. ***ftp_client_command.cpp:*** *ftp_client_control.hpp, ftp_client_ui.hpp, ftp_server_response.hpp, ftp_client_command.hpp*

iii. ***ftp_client_connection.cpp:*** *ftp_client_connection.hpp*

iv. ***ftp_client_control.cpp:*** *ftp_client_connection.hpp, ftp_client_ui.hpp, ftp_client_control.hpp*

v. ***ftp_client_ui.cpp:*** *ftp_client_ui.hpp, ftp_client_command.hpp*

7. **Organize your code nicely** and **add adequate comments** on your code.
8. **Build** your FTP Client software using **make**.
9. **Copy** this **Test FTPServer Software** in your **bin** folder.
10. **Run** the copied **FTP Server** typing

    >***bin/myftpserver 2019***

    This will run the FTP Server software in your computer and the server is now waiting for the client connection at port 2019.

    Note that this FTP Server Software is built and tested only in Linux Debian machines of the lab. Run this software in other machines at your own risks.

11. **Run** your FTP Client software typing

    >***bin/myftp 127.0.0.1 2019***

    Above command will connect your client software to an FTP Server software running at your machine and listening for client connection at port ***2019.***

12. **Test** all the commands in your FTP Client software against the connected FTP Server.

    You can download this example FTP Client software in your **bin** folder and run it to get an idea what is expected from you.

13. Once your project is completed and tested successfully, make **ftpclient.tar.gz** of your project. In order to make the tar ball, you need to move up your project's parent directory and type the following:

    >**tar –zcvf ftpclient.tar.gz FTPClient**

14. Submit your **ftpclient.tar.gz** and demonstrate your software in the lab before the deadline. Submit a **printed copy** of all your cpp files to your instructor. You need to submit your **ftpclient.tar.gz** and **printed copy** of the **cpp** files before your final demonstration.
15. Submit the filled up **contribution form** if you have done your project in a partnership with another student.

Partnership

You are allowed to complete this project with another partner. In that case, your contributions must be marked in your source code. If you are the only contributor of a function, you should mark yourself as the sole author of that function before the function header. If you are the only contributor of all the functions in a file, you should mark yourself as the sole author of the file at the beginning of the file. If both you and your partner have contributed to write a function or all the functions of a file, both of you should mark yourselves as the authors at appropriate position. You and your partner must complete, sign, and submit this **contribution form** to your instructor along with the **printed copy** of all your source codes (**cpp** files only).

Deadline and Submission

The deadline to demonstrate and submit the project is **October 22, 2019, Tuesday by the lab hour.**

The submission mechanism will be announce later before the deadline.

Evaluation

Demonstration	**Commands**	**Marks**
	help	**5**
	user	**5**
	pass	**5**
	pwd	**5**
	dir	**20**
	cwd	**10**
	cdup	**10**
	get	**20**
	quit	**5**
Code Quality and Comments		**15**
Total		**100**