CMPE 156/L, Winter 2018

Programming Project Due: ___Mar. 20____

The purpose of this project is to develop the server and client sides of a simple ftp application. Refer to the FTP RFC for the protocol details: https://www.ietf.org/rfc/rfc959.txt.

Requirements

The server code must accept its listening port number as the first command line parameter (from argv[1]). That is, you should be able to run the server using the command line

./ftpserver <listen-port>

FTP commands

The server should understand and process the following FTP commands:

Protocol Command	Description
LIST filelist	List files or directories
ABOR	Abort pervious FTP command and any data transfer
PORT n1,n2,n3,n4,n5,n6	Client IP address(n1.n2.n3.n4) and port (n5*256+n6)
QUIT	Exit program
RETR filename	Retrieve (get) a file
STOR filename	Store (put) a file

FTP replies

Each FTP command generates a one-line reply from the server. For example, the QUIT command could generate the reply:

```
221 Goodbye.
```

Implement all necessary replies. Other typical replies:

- 200 Command OK.
- 214 Help message (for human user).
- ~~331 Username OK, password required.~~
- 425 Can't open data connection.
- 452 Error writing file.

- 500 Syntax error (unrecognized command).
- 501 Syntax error (invalid arguments).

FTP interface
The client code start and gives a user interface prompt waiting for user commands.

./ftpclient <server-ip> <server-listen-port>
ftp> <client-cmd> [<cmd-arg>]

Interface Command	Description
ls [<name>]	List current directory or filename
get <filename>	Get the file
put <filename>	Upload the file
quit	Exit the user interface

What to submit?
You must submit all the files in a single compressed tar file (with tar.gz extension). The files should include:

1. All source code necessary to build and run the client and server.
2. A Makefile that can be used to build the client and server binaries.
3. Any test files you used to test your design.
4. Documentation of your design in plain text or pdf. Do not include any Microsoft Word files. The documentation should describe how to use your application and the internal design of your client and server implementations.
5. A README file including your name and a list of files in the submission with a brief description of each. If your code does not work completely, explain what works and what doesn't or has not been tested.
6. Organize the files into directories (src, bin, doc, etc.)

Grading
Each submission will be tested to make sure it works properly and can deal with errors. Grades are allocated using the following guidelines:

Basic functionality:	20%
Basic testing	20%
Dealing with errors	30%
Documentation	15%
Style/Code structure, etc.	15%

The files must be submitted before midnight on the due date.

Honor Code
All the code must be developed independently. All the work submitted must be your own.

Appendix

Connection Management

There are three uses for the data connection.

- Sending a file from the client to the server.
- Sending a file from the server to the client.
- Sending a listing of files or directories from the server to the client.

For example, the FTP server sends file listings back across the data connection, rather than the control connection. The control connection stays up for the duration of the client–server connection, but the data connection can come and go, as needed.

The end-of-file is denoted by closing the data connection. This implies that a brand new data connection is required for every file transfer or directory listing. The normal procedure is as follows:

How are the port numbers chosen for the data connection, and who does the active open and passive open?

- The creation of the data connection is under control of the client, because it's the client that issues the command that requires the data connection (get a file, put a file, or list a directory).
- The client chooses an ephemeral port number on the client host for its end of the data connection. The client issues a passive open from this port.
- The client sends this port number to the server across the control connection using the PORT command.

The server receives the port number on the control connection, and issues an active open to that port on the client host. The server's end of the data connection always uses port <listen-port-1>.

Error Handling

The client and server should handle error conditions without crashing or hanging. For example, if a file or path doesn't exist, an appropriate message and code should be sent back and shown to the user. Assuming filename 'foo' doesn't exist, there should be an error code of 450 and a message about it.

```
ftp> ls foo
450 foo: No such file or directory
ftp>
```