CSCI 452: Network and Web Programming

TCP Socket Project

Due February 1st, 3 PM

- Submit your assignment at http://www.networks.howard.edu/lij/courses/submit_hw.html
- Double check your submission. (You may submit multiple times with the latest one considered for grading.) No correction will be allowed after deadline even if wrong files are submitted.
- Create a project on Github and commit your code periodically. Make your project private initially and make it public at the deadline. (To create a private repository for free, sign up with your .edu email address.)
- Materials to be submitted (Zip all the materials and submit the zip file)
  a. C code of the client
  b. C code of the server
  c. A PDF file that contains the following information:
     - A URL of your project on Github.
     - A screenshot of the Github web page that shows the summary commit history of your project. The commit history is expected to show the progress of your development over time. There must be at least 5 commits for the program(s) of each side (client and server). While you are developing, at least one commit must be done every two hours. More is better.
     - The details of each commit, which can be obtained from Github web site. Each commit should contain non-trivial modifications. If there are much more than 5 commits for the program(s) of a side, select five significant commits that are spread evenly over time.
- Grading

No credits will be given and the submission will not be evaluated further if any of the following is true.

- Any of the required files has a wrong format.
- Any of the required files is missing.
- The code file has more code than what is required.
- The commit history does not satisfy the requirements.
- The commit history shows suspicious practice.

After that, programs will be judged according to the following criteria.

a. Indentation: 5%
b. Comments in code: 10%

c. Correction of programs: 85% (judged by the outcome of tests)

In this project, you will develop two network programs in the C language by modifying the echo examples, one for client and one for server.
The client program keeps reading input from user and performs actions as the following:

- If the user enters 's', it will ask user to enter a string. It then sends "CAP\nxxx\n" to the server (where "xxx" is the string entered by the user), and receives a message from the server and output the message on the screen.
- If the user enters 't', it will ask user to enter a string. It then sends "FILE\nxxx\n" to the server (where "xxx" is the string entered by the user), and receives data from the server and saves the data to a file named "xxx" in the directory where the client program is in.
- If the user enters 'q', it will exit.
- If the user enters anything else, it will ignore.

The server program performs the following actions:

- If "CAP\nxxx\n" is received from the client, it capitalizes the received string and sends "###\nCCC" back to the client, where "###" is the number of characters in the capitalized string and "CCC" is the capitalized string.
- If "FILE\nxxx\n" is received from the client, it reads the content from the file named "xxx" from the directory where the server program is in and sends "###\nDDD" back to the client, where "###" is the number of bytes of the file and "DDD" is the content of the file. Note that the file may contain binary data such as images. If no file of the name "xxx" exists, it sends "###\nNOT FOUND" back to the client, where "###" is the number of characters in the string of "NOT FOUND".

The client should be invoked by the following command:

```
<client> <server IP> <server port>
```

Where `<client>` is the name of the client executable file name, `<server IP>` is the IP address of the server, `<server port>` is the TCP port of the server.

The server should be invoked by the following command:

```
<server> <port>
```

Where `<server>` is the name of the server executable file name, `<port>` is the port the server listens to.