CMSC 257 Assignment 6: Sample ftp client

Project due date: 11:59 pm EST, 05/05/17

Client-Server Set-up

- The goal of this project is to create two programs: a client and a server. The server will operate in a sequential mode in order to transfer files to the client.
- For each connection from the client, the server will read a single file name from the client and then send the file back to the client.
- The file name that the client would like the server to fetch is specified in the command line of the client program. To execute the client, you need to use:
  ./client sample_file.c
- The server needs to be invoked by simply typing in: ./server

Code set-up

- Create two directories named "Server_dir" and "Client_dir" under the main assignment directory "Assignment_5" and store the server program named "server.c" and few files that the client might ask for from the server like "add.c", "hello.c", etc. The client program "client.c" is saved in the directory "Client_dir". In order to run the implementation the "server.c" is executed first and then in another terminal window the "client.c" is executed with the file name in the command line that the client would like the server to fetch.

- As both the server program "server.c" and client program "client.c" will be executed on the same machine, assign the IP address as "127.0.0.1" which is a default IP for the current machine.

Systems calls to be implemented:

- Client
  - socket()
  - htons
  - connect()
  - read/write

- Server
  - socket()
  - htons()
  - bind()
  - listen()
  - accept()
  - read/write

Requirements:

- Each read/write into sockets needs to use a buffer of 50 characters. Note that the file-name to be transferred should fit the 50 characters requirement as well.
- The server.c should have an infinite loop to cater to multiple client requests (sequentially).

- The server should use file handling to open the file and start sending chunks of bytes to the client.
- The client.c code needs to use an infinite while loop to keep reading in the contents of the file. As the file is not present, it needs to create a file in “w” mode and write components to it as and when it receives chunks from the server.
- Use a special string: “cmsc257” as a terminal string; the server sends this string at the very end to tell the client that the file has been transferred. The client, on receiving the string, needs to exit gracefully.
- Use a signal handler to “gracefully” terminate the server on Ctrl-C; existing client connections should be terminated (by stopping the data transfer or waiting for the transfer to complete) before the server exits.
- Use fork on the server to allow multiple simultaneous file downloads from different clients; you may want to specify the maximum number of client connections that can work in parallel.
- Use PORTs 2000-3000 for setting up your sockets.
- Submit a tar file as before with your client/server directories and corresponding files within. Note that you need to turn in TWO versions of the server.c code: (i) first with the sequential server version and (ii) forked version that can handle multiple client requests simultaneously.

Note: Late assignments will lose 5 points per day upto a maximum of 3 days. Code must be submitted in the prescribed format.

For questions on grading, contact Syed Khajamoinuddin < lnusk@mymail.vcu.edu >
