CS 372 Introduction to Computer Networks Spring 2014

Programming Assignment
Due Sunday, May 18th by 11:59 pm.
Submit the source code files at http://engr.oregonstate.edu/teach AND Blackboard.

Objectives:

1. Implement 2-connection client-server network application
2. Practice using the *sockets* API
3. Refresh programming skills

Program Plan:

You must submit a typed Program Plan for the Project by 11:59pm Sunday, May 4th 2014. This is NOT OPTIONAL. ***If you do not submit the Program Plan on time, you will receive ZERO credit for the project.*** This Plan must include the following:

- Name/etc
- Pseudocode for the client and server
- Written explanation of functionality for client and server

Program Requirements:

- Of course, your program **must be well-modularized and well-documented**.
- Your programs must run on an `engr` server. (Please specify which server in your documentation)
    - Probably the best way to do this is to use SSH Secure Shell, Putty, or another terminal emulator to log onto *access.engr.oregonstate.edu* using your ENGR username/password and note which flip you get (I believe there is flip1, flip2, flip3). It will be easiest if you bring up two instances of the shell on the same flip server and use one to run the server, and the other to run the client (this is how I will be testing!).
- You may not use *sendfile* or any other predefined function that makes the problem trivial.
- Your program should be able to send a complete text file. You are not required to handle an "out of memory" error. Separate grading for short text files and long text files.
- Use the directories in which the programs are running. Don't hard-code any directories that might be inaccessible to the graders.
- If you use additional include-files or ***make-files***, be sure to hand them in with your program.
- Be absolutely sure to cite any references and credit any collaborators.

The Program:

Design and implement a simple file transfer system, i.e., create a file transfer server and a file transfer client. Write the *ftserve* and the *ftclient* programs, and submit the two source code files. Write the program in C, java, or python! Include any pertinent makefiles and include instructions for executing the program in the program description comments. The final version of your programs must accomplish the following tasks:

1. *ftserve* starts on host A, and validates any pertinent command-line parameters (port, etc).
2. *ftserve* waits on port 30021 for a client request.
3. *ftclient* starts on host B, and validates any pertinent command-line parameters. (host, port, command, filename, etc…)
4. *ftserve* and *ftclient* establish a TCP control connection on port 30021. (For the remainder of this description, call this connection *P*)
5. *ftserve* waits on connection *P* for *ftclient* to send a command.
6. *ftclient* sends a command (`list` or `get <filename>`) on connection *P*.
7. *ftserve* receives command on connection *P*.
   a. If *ftclient* sent an invalid command, *ftserve* sends an error message to *ftclient* on connection *P*, and *ftclient* displays the message on-screen.

   otherwise
   - *ftserve* initiates a TCP data connection with *ftclient* on port 30020. (Call this connection *Q*)
   - If *ftclient* has sent the `list` command, *ftserve* sends its directory to *ftclient* on connection *Q*, and *ftclient* displays the directory on-screen.
   - If *ftclient* has sent `get <filename>`, *ftserve* validates `filename`, and either
     - sends the contents of `filename` on connection *Q*. *ftclient* saves the file in the current default directory (handling "duplicate file name" error if necessary), and displays a "transfer complete" message on-screen

     or
     - sends an appropriate error message ("File not found", etc.) to *ftclient* on connection *P*, and *ftclient* displays the message on-screen.

   b. *ftserve* closes connection *Q*.
8. *ftclient* closes connection *P (don't leave open sockets!)*.
9. *ftserve* repeats from 2 (above) until terminated by a supervisor *(don't leave open sockets!).*

Options:

There are many possibilities for extra credit (all extra credit must be document and referenced), e.g.

- Make your server multi-threaded.
- Implement username/password access to the server.
- Allow client to change directory on the server.
- Transfer files additional to text files.
- etc.

Notes:

- *Beej's Guide* will be helpful. It has everything you need for this assignment.
- Don't hard-code the port numbers
- We will be testing this with single commands at a time, for example
  `ftclient flip2 30021 list` (this will list the directory in the server and then terminate)
  `ftclient flip2 30021 get shortfile.txt` (this will download shortfile.txt then terminate)
  `ftclient flip2 30021 get shortfile.txt` (this will get an error, as this file will already exist in the client directory)
- Don't use the well-known FTP port numbers, as these will already be in use by. Use the port numbers specified in the description to avoid conflict with reserved ports.
- When testing, don't use 30021 or 30020, as these will be in use by other students.
- We will test your system with text files only (unless your documentation specifies additional file types), one very large and one small.
- If you implement extra credit features, be sure to fully describe those features in your program documentation, or you won't receive any credit.
- Programs will be accepted up to 48 hours late with a 10% penalty.