COSC4653 – Advanced Networks

Assignment #4

FTP Server/Client (in C)

Due: March 2, Midnight

Format

Submit a single zip file containing your two C source code files. Do not submit any other files.

Name your zip file Baas4.zip, where "Baas" is your last name and "4" is the assignment number. Submit the zip file by way of the COSC 4653 Assignment #4 link on Blackboard.

Your C programs should compile, link and run with no error messages.

This is an individual assignment. Do not exchange your solution with other students in the class.

Assignment Objectives

- Practice implementing a networking program in C
- Practice making a TCP connection between a server process and a client process, each running on different computers
- Practice sending multiple bytes of information from a server process to a client process over the network by way of multiple read and write operations

Assignment Summary

In this assignment, you will develop two programs, a server program and a client program. These programs shall be based on the sample day-time client and server programs described in the Stevens textbook. You shall modify those two programs so that the server will act as a simple ftp server, and the client will act as a simple ftp client. Both programs shall be written in standard C using none of the wrapper functions illustrated in the Stevens book.

Assignment Directions

Download the two source code files named `day-time-tcp-client-modified.c` and `day-time-tcp-server-modified.c` from the assignment page on Blackboard.

Rename these two source code files to `ftp-client.c` and `ftp-server.c`, respectively.

Place a completed comment header block at the top of each file.

Only make your source code modifications in the `main()` function of each file.

In your server program, modify the algorithm so that the source code reads the contents of a text file line by line from standard in and sends these contents to the first client that connects to it.

When the file is completely sent, have the server terminate normally. Remove any source code in the `main()` function that was used specifically for the day-time functionality.

In your client program, modify the algorithm so that it reads a series of lines sent by the server, printing each line to standard out as it is received, until a buffer is read that has a length of zero. Then have the client terminate normally. Use the `MAX_LINE_LENGTH` constant for the size of the buffer. Remove any source code in the `main()` function that was used specifically for the day-time functionality.

When testing your client and server programs, note that the only things that they know about each other are the network connection and the steps for exchanging the file data (i.e., our FTP protocol); consequently, you may ask another student in the class to start up his server process so you can contact it with your client process. Do the same for his client process and your server process.

Sample Run of Client and Server Programs

First start the server program in a command shell on one computer as shown below:

```
% ftp-server < sourceFile.txt
```

Then start the client program in a command shell on another computer as shown below, where the IP address entered on the command line is the address of the computer where the server program is running:

```
% ftp-client 1111.2222.3333.4444 > targetFile.txt
```

After both programs terminate, the contents of the target file should exactly match the contents of the source file.

Design and Implementation Constraints

Follow the same design and implementation constraints listed for Assignment #1. This includes, for example, such items as the comment block, the naming of constants, the naming of variables and functions, and the placing of a comment after each right curly brace (e.g., } // End if).

Check the status value returned by each networking and file function that is called in your source code. If an error is detected, print an error message using `perror()` and then call `exit()`.

This assignment is used with permission from Dr. Jay-Evan Tevis.