CSE 5462: Lab2 (100 points)
File Transfer Program – Stream Sockets

Demo in Class or by appointment: January 23 (Thursday)
Electronic Code Submit Deadline: 9pm, January 23

Important: You will be told to write either a client or a server. TEST with 2-3 people writing the reciprocal component.

We will demo in class or in lab. For in class demos, we will go around the room. Be prepared to demo at the start of class. The submitted code will be used only to verify that you did not copy from others, to compile and re-run your program, to make sure you were indeed demonstrating your own code, and to grade for documentation of your code. You may be asked to connect your client/server to a client/server from another pair of students!

- For socket programming use can either use the CSE linux system (stdlinux) or your own machines. For your own machines, demo can be done in classroom.

Write a file transfer application using TCP STREAM oriented sockets. The file-transfer assignment will include a server called ftps and a client called ftpc. (you only write one of these!)
You will first start the server using the command

ftps <local-port> You will then start ftpc with the command
ftpc <remote-IP> <remote-port> <local-file-to-transfer>

The ftpc client will send all the bytes of that local file. The ftps server should receive the file and then store it. Make sure that the new file created by ftps is in a directory called recvd, to avoid over-writing the original file. You may NOT read the entire file into memory. You should choose some buffer size on the order of a few 1000 bytes on both the sender and receiver sides.

PROTOCOL:Client will send an initial 4-byte header to tell the server the size of the file about to be sent. The server will, after reading the entire file, send back a 4-byte response telling how many bytes were actually sent.

The next 20 bytes (blank fill on right end if less than 20 bytes) will be the filename.

Note that your program should work for a binary file (images etc.) of any arbitrary size. Do not use ascii functions (e.g., strcpy, scanf, sscanf) to manipulate binary data. If you want, try these functions to see what happens. The client and server should be running on different machines on stdlinux (e.g., epsilon, zeta, beta, delta, gamma).

You can use the sample images on the project webpage to test your program. You can use “diff” to ensure that the transferred file is the same as the original one.

Submit well-documented and well indented code along with a README file explaining how to run
the program, and a makefile. Submit it using GitHub, in a subdirectory called Lab2

You may need to allocate enough time (often quite difficult to estimate) for debugging. Follow the exact specifications and requirements as outlined in this document.
Your program does not need to support transfer of multiple files. It is ok if it quits after transferring one file.

The grading rubric is as follows:

1. Program correctness and robustness: 70%– Program MUST work to get these points
2. Coding style (e.g., comments, indentations, subroutines): 20%
3. Documentation (the README file): 10%

What you should get from this lab?
Tools:

- basic socket IO (socket(), bind(), connect(), listen(), send(), recv(),close(), getsockname() etc);
- network order functions (ntohl, ntohs, htnol, htnol)
- basic string manipulation (memcpy(), memset(), etc);
- trimming data received on a socket connection (issspace(), etc)
- acks – application level

Concepts:

- what is a socket;
- how is it similar/different from a file;
- how do you make sure a port is in the right network order;
- how do you pass integers between machines;
- what are the characteristics of a stream oriented socket, how is similar/different from a file;
- how does the receiving application know how much data to expect;
- how does the sending application know that all the data was received *by the application*;