Programming Assignment 1: Socket Programming

In this assignment, you'll write a client that will use sockets to communicate with a server that you will also write. Here's what your client and server should do:

Your client should first accept an integer between 1 and 100 from the keyboard, open a TCP socket to your server and send a message containing (i) a string containing your name (e.g., "Client of John Q. Smith") and (ii) the entered integer value and then wait for a sever reply.

Your server will create a string containing its name (e.g., "Server of John Q. Smith") and then begin accepting connections from clients. On receipt of a client message, your server should

i. print (display) the client's name (extracted from the received message) and the server's name
ii. itself pick an integer between 1 and 100 (it's fine for the server to use the same number all the time) and display the client's number, its number, and the sum of those numbers
iii. send its name string and the server-chosen integer value back to the client
iv. if your server receives an integer value that is out of range, it should terminate after releasing any created sockets. You can use this to shut down your server.

Your client should read the message sent by the server and display its name, the server's name, its integer value, and the server's integer value, and then compute and the sum. The client then terminates after releasing any created sockets. As an aside (and as a check that you are doing things correctly, you should make sure for yourself that the values and the sums are correct!)

You've got the basic assignment done. The last part of the assignment should be a fun, social thing to do. Team up with someone from the class (advertise on the Piazza class list if you are looking for a partner) and get your client to interact with their server or vice versa.

You should program your client and server to each print an informative statement whenever it takes an action (e.g., sends or receives a message, detects termination of input, etc.), so that you can see that your processes are working correctly (or not!). This also allows the TA to also determine from this output if your processes are working correctly. You should hand in screen shots (or file content, if your process is writing to a file) of these informative messages as well as the required output of the client and server (name strings, integer values and sums).

Rewrite your server to be a concurrent server - that is a server that waits on the welcoming socket and then creates a new thread or process to handle the incoming request (i.e., carry out the username verification and message-of-the-day protocol described above).

Programming Languages and Operating Systems

You may write your programs in Java, C++, Python, or C on any platform (Liux/unix, Mac, PC) you want; but ask me first.

Programming the assignment in Python.

A Python socket tutorial is http://docs.python.org/howto/sockets.html

Programming notes

Here are a few tips/thoughts to help you with the assignment:

- You must chose a server port number greater than 1023 (to be safe, choose a server port number larger than 5000). If you want to explicitly choose you client-side port, also choose a number larger than 5000.
- You may need to know your machine's IP address, when one process connects to another. You can telnet to your own machine and seeing the dotted decimal address displayed by the telnet program. You can also use the UNIX nslookup command . On Windows, see the ipconfig utility. On a Mac, you can run the terminal program and use the ifconfig commend (just type in `ifconfig` or `ifconfig | grep "inet ")`.
- Make sure you close every socket that you use in your program. If you abort your program, the socket may still hang around and the next time you try and bind a new socket to the port ID you previously used (but never closed), you may get an error. Also, please be aware that port ID's, when bound to sockets, are system-wide values and thus other students may be using the port number you are trying to use.

Here's a page that will get you started that will get you started in Python: http://ilab.cs.byu.edu/python/threadingmodule.html and http://www.eurion.net/python-snippets/snippet/Threaded%20Server.html.

If you find better pages for C, Java or Python, please share with the class!