Assignment 1

TCP Client-server
Due: August 3, 2009, 1-00 PM

1. Write a simple TCP iterative server and client to evaluate arithmetic expressions. The client-server system has the following functionalities:

    a. The client asks for an arithmetic expression from the user. The user enters an arithmetic expression containing only positive integers and the binary operators +, -, *, and /. The arithmetic expression can be at most 100 characters long. There may or may not be spaces before and after the operators. For example, all of the following are valid expressions to enter: "13 + 42*5", "10+2/4", "5 + 6 - 3". Assume that all operators have the same precedence and are left-associative.

    b. The client sends the expression to the server. ***You should send only the number of bytes needed to send the expression properly and not send the complete buffer if not needed.***
    c. The server computes the expression and sends the integer back to the client.
    d. The client displays the result on the screen. It then prompts the user to enter the next expression.
    e. The client terminates when the user enters a –1.

2. Write a TCP concurrent server and client to do the above. Note that more than one client can connect to the server at the same time.

You can assume that the user always enters a valid expression. To test your program, first run the client and the server on the same machine. Use the IP address 127.0.0.1 and the port no. = 50000 + last 4 digits of your roll no. (for ex., if your roll no. is 06CS1001, use port no. = 50000 + 1001 = 51001). Finally, test your program by running the client and the server on two different machines (set the server IP address appropriately).

You should submit three C files, the client, the iterative server, and the concurrent server. Name the files *client.c*, *server_it.c*, and *server_conc.c* respectively.

Things to note (not part of assignment):

1. Print out the return values from the *send* and the *recv* calls. Look up the man page to see what they mean. In general, read the man page to understand the *send/recv* (*write/read*)

calls thoroughly.

2. Write a small program in which the client waits for the server to send something. Server does not send anything, but just sleeps for 30 seconds (enough time for you to start the client after starting the server) and then closes the socket. What do you see happen at the client?
3. Based on the above, and your general knowledge of programming, can you now devise a scheme in which the client and the server can send each other arbitrary length data (you cannot assume any maximum length), **but you cannot use any buffer of size more than 100**.