CS4516 Advanced Computer Networks C13

Program 0 8 points

Enhanced TCP Echo Server and Client
Due: Monday, January 14, 2013 at 11:59 p.m.

This assignment provides all students an opportunity to review the standard procedures for developing, executing and testing C socket programs on a Linux system. The task is to start from any published version of the TCP Echo Client and TCP Echo Server that you prefer. It can be essentially a **copy** of these two programs in the D&C textbook, in any other textbook or from the class Powerpoint slides (your comments must indicate the source). The procedures are:

1. Write the client program and the server program on a CCC Linux machine using your favorite editor.
2. Create and use a **make** file to build, test and debug both programs.
3. Create a **README** plain text file to assist in the grading of program 0.
4. Tar all source program pieces, the make file(s) and the **README** file into a single tar file for submission.
5. Use the Unix version of ***turnin*** to submit the tarred file for grading.

The assignment is to start from a published **basic** TCP Echo Client and TCP Echo Server code and develop an **enhanced** Echo Client and Server in C or C++ using Linux socket commands. Note: this assignment is intended to be an **easy** assignment for all those who have completed CS3516 and done multiple socket programming assignments. The **enhanced** Echo client and Echo server implement an ***echo protocol*** while running on different CCC machines and communicating with each other using TCP.

The Enhanced Echo Client

The **basic** Echo client connects to the **basic** Echo server and sends its data (a single string) to the server. The data that the **basic** Echo client sends is a string provided as the second client command-line argument. The **basic** Echo client prints the single string of data sent back by the **basic** Echo server.

The command line for the **enhanced** Echo client has the form:

%**my_EnhancedEClient ServerMachine "string 1" "string 2" "string 3" "string 4" "string 5"**

where '**my**' **must be** replaced by the initials of the program author**.**

The following is the output from the **enhanced** Echo client given the above command line:

EnhancedEClient received: string 1
EnhancedEClient received: string 2
EnhancedEClient received: string 3
EnhancedEClient received: string 4
EnhancedEClient received: string 5
EnhancedEClient: done

The **enhanced** Echo client receives the name of the computer where the server is running (e.g. CCCWORK2) as its first command-line argument **ServerMachine** and uses a Linux system call to convert the server name to the associated server IP address.

The other enhancement for both the **basic** Echo client and the **basic** Echo server is that the number of strings to be echoed varies from one to five. The **enhanced** Echo client sends a series of data strings (one per TCP packet) to the **enhanced** Echo server. The **enhanced** Echo client prints out each data string sent back by the **enhanced** Echo server. Once the **enhanced** Echo client has sent and printed all the echoed strings, it sends one additional 'delimiter' TCP packet containing the two ASCII bytes **DLE ETX**. Once the **enhanced** Echo client receives the echoed **DLE ETX** bytes, it prints out a done message, disconnects and terminates.

The **enhanced** Echo client accepts strings of length 1 to 12 bytes inclusive and prints out an error message for any out-of-range input string that is too long. Any out-of-range string is not transmitted, echoed or printed. The **enhanced** Echo client then processes the next string (if any).

The Enhanced Echo Server

After connecting to the **enhanced** Echo client, the **enhanced** Echo server (which is started first) runs in a loop accepting up to five strings from the **enhanced** Echo client as up to five TCP packets. When the **enhanced** Echo server receives a packet containing the two ASCII bytes **DLE ETX,** it echoes this packet back to the client and prints out a message of the form:

%EnhancedEServer echoed **n** strings.

where **n** is the number of strings echoed by the server prior to receiving the **DLE ETX**.

The **enhanced** Echo server then disconnects and terminates.

Program 0 turnin specifications

Turn in Program 0 using the ***turnin*** program on the CCC machines. You must turn in a tarred file that includes: your source code files, a **make** file, a **README** file and a sample output file from a test run. The **make** file(s) should include the ability to cleanup leftover output and intermediate files between compile and execution cycles. The **README** file provides any information to help the TA test and grade your **enhanced** Echo Client and Echo Server on separate CCC machines.