CS 39006: Assignment 1

Using TCP Sockets

Date: 10-Jan-2022

Deadline: 17-Jan-2022, 2 pm

Problem 1(a):

In this assignment, you will write a client-server system using TCP sockets. Specifically, you will write two programs, a TCP client program and an **iterative** TCP server program. The client program will send a text file to the server program. The server will count the number of sentences, words, and characters in the file and send them back to the client. The client will then display it. The sequence of operations to be followed are shown below.

1. The server opens a TCP socket and waits for a request from the client.
2. The client reads the name of a text file from the user as a command line argument. It then looks for the file in the local directory. If the file is not there, it prints a suitable error message ("File not found") and exits. If the file is found, the steps below are executed. You can assume that the file name has maximum 100 characters and the file, if found, will always be a text file containing English sentences.
3. The client establishes a connection to the server using the connect() call.
4. The client reads the contents of the file and sends it to the server using the TCP socket. Since the file can be arbitrarily long, the client cannot send the entire file in a single send() call, and sends the file in small chunks using multiple send() calls until the entire file is transferred. **The chunk size used by the client is not known to the server.**
5. The server receives the file, and computes the number of characters, number of words, and number of sentences in the paragraph, and sends these numbers back to the client using the same TCP socket. The server then waits for the next client request.
6. The client will print these numbers on the screen.
7. You can assume that in the file, each sentence is terminated with a full stop character only, and the fullstop character appears only at the end of sentences and nowhere else. **Also, any sentence has at least one character which is not a fullstop.**

Your code should satisfy the following conditions:

1. The maximum size of any buffer used by either the client or the server is 100 bytes (you can assume this is known to both client and server).
2. The client does not determine or send to the server the size of the file in any way.
3. You cannot use fopen/fscanf functions. You must use open/read functions to open and read the file.
4. The server is not allowed to write anything in a file. It should count on the fly from the buffer.

You can assume that only a single client will connect to the server at any one time.

You have to submit two C programs – my_tcpserver.c (containing the server program) and my_tcpclient.c (containing the client program). Create a single compressed folder (zip or tar.gz) with the name <roll number>_Assgn1a.zip or <roll number>_Assgn1a.tar.gz, and put both your client and server programs in it. Submit this compressed folder in Moodle in the proper link by the deadline.

Problem 1(b):

Implement the above using an iterative UDP server. For this part, assume that no errors will happen even though UDP is unreliable, we just want you to exercise the UDP calls. (However, do think what will problems can happen if messages get lost and how you may still do it correctly, it has the basis of a technique we will read later. Not required for the assignment, just suggestion to think).

You have to submit two C programs – my_udpserver.c (containing the server program) and my_udpclient.c (containing the client program). Create a single compressed folder (zip or tar.gz) with the name <roll number>_Assgn1b.zip or <roll number>_Assgn1b.tar.gz, and put both your client and server programs in it. Submit this compressed folder in Moodle in the proper link by the deadline.