MINOR ASSIGNMENT-05

UNIX Network Programming (CSE 4042)

Publish Date: 24-06-2022 **Submission Date:** 04-07-2022

TCP Multi-Client Server and Concurrent Server

This assignment is designed to practice with multiple clients communicating with the server.

1. Write a program to create two TCP clients and a TCP server. The server reads two numbers and sends one number to each client. All the clients will display the received number from the server. The server will display the port numbers of the connected clients.

2. [**Exchange of numbers between two clients:**] Create a program to connect two TCP clients to a TCP server. Both the clients read 10 numbers and send the numbers to the server. The server sends the received numbers to opposite clients. The clients will display the numbers.

3. A TCP server reads 10 pairs of numbers [scanf("%d%d",&x,&y);], where x is 1 or 2. If x is 1 then y is send to first client. If x is 2 y is send to second client. Both clients display the numbers, which they have got.

4. Write a program to create two servers and one client. Initially the client reads two numbers. These are port numbers of the servers. Then the client reads 2 more numbers **P** and **R**. The number **P** is sent to one server and the number **R** is sent to another server.

5. Write three TCP-socket programs: **server.c**, **clienta.c**, and **clientb.c**. These programs are executed from three different windows. The **clienta** reads array of n numbers and sends to server. The server will compute the number of occurrence of each element and sends to **clientb**. The **clientb** will add each occurrences and displays the sum.

6. Write a TCP-socket program to create a server and a client. The client will read an array of n numbers and sends the half of the array of numbers to the server. The server will sort half of the array and sends back to the client. Client will sort the second half of the array of numbers and merge the both sorted halves into a single sorted array. The server will display the first sorted half and the client will display the whole sorted array.

7. Write three TCP-socket programs: **server.c**, **clienta.c**, and **clientb.c**. These programs are executed from three different windows. The **clienta** reads a n digit number and sends to the server. The server will reverse the received number and sends the to the **clientb**. **Clientb** will display the received number and test whether the received reverse of the original number is odd or even. Then, the **clientb** sends even or odd back to server. The server will displays the result of even or odd computed by **clientb**.

8. Write a TCP-socket program to create a server and two clients. The client will read a $m \times n$ matrix and sends it to the server. The server will read a $m \times n$ matrix also. The server will add both the matrices and sends back to the seond client. The second client will display the original matrix and sum of both the matrices.

9. [**Check if a decimal integer is a palindrome:**] Design a TCP client serevr program to connect one client to two servers. The first server will send a decimal integer to the client and the client will check whether the number is **palindrome** or not. The result of YES/NO for palindrome checking will be forwarded to the second server. The second server will display PALINDROME or NOT PALINDROME.

10. Design a client-server program for one server and two clients. The server takes a double x and an integer y. If y is found to be negative, the first client will compute $\frac{1.0}{x^y}$ otherwise the second client will compute x^y. Finally the server will display the reslut of x^y. Try to ignore overflow and underflow.

11. Design a concurrent TCP client/server program using **`fork()`** for an echo server that performs the following steps: [Refer chapter 5 of your text book for TCP client/server example]

   i) The client reads a line of text from its standard input and write the line to the server.
   ii) The server reads the line from its network input and echoes the line back to the client.
   iii) The client reads the echoed line and prints its on its standard output.
   iv) The server will keep a track of the client number that is connected to the server and also the child server process ID.

   After doing all the above steps, meet the following cases:

   a) Start the server on the host assuming server never terminates. Before starting the client, run the **`netstat`** or **`ss`** command to verify the state of the server's listening socket, local protocol address and foreign(or peer) protocol address. Attah the screenshot of the command output
   b) Now, start the client specifying server's protocol address (i.e IP and Port) and ensures that the connection is established. Run **`netstat`** or **`ss`** command to show additional lines of output corresponding to the TCP connection, **ESTABLISHED**. Attached the screenshot.
   c) Use the **`ps`** command to check the status and relationship of these processes. Attach the screenshot of the command.
   d) Observe that after connection is established, whatever you typed to the client is echoed back. Now type terminal EOF character (CTRL+D) to terminate the client. Try to execute **`netstat`** or **`ss`** command immediately to look into `TIME_WAIT` state and verify that listening server is still waiting for another client connection.
   e) Use **`ps`** command after the client terminate to verify that the child server enters into `ZOMBIE` state. Attached the screenshot showing the state symbol `Z` for child-server.
   e*) [**Optional**] You may add signal handler that will call `wait` or `waitpid` to avoid the case of `ZOMBIE` around as they take up space in the kernel and eventually you can run out of processes. Whenever you `fork` children, you must wait for them to prevent them from becoming zombies.
   f) Observe the situation that will happen, when the server terminates before the client. Does it notify to the client?
   g) Observe the case, if the client will start before the server.
   h) Create a new client code as like the previous client to establish 10 connections to the server.

You are required to write the sample input and output for various part of the assignment.