CN Lab Assignment 3

Max Marks:20 **Deadline : 20-04-2023**

Instructions:

● **Please submit original work. Plagiarism/copying of any form, will generate zero mark. Plagiarism from the internet also should less than 8%.**

● Submit all documents in google classroom.

● Save all submission files into a single folder and submit the compressed folder.

● **Explain the working of functions in proper comments.**

● **There are no partial marks for task1 and task3.**

● **All tasks required proper executable codes, screenshots for otput etc.**

● **No compile time and run time errors are entertained.**

● **Create a readme file explaining how to execute your code. (Terminal commands etc.).**

Must follow the following naming convention:

○ Name the ZIP file submission as,

Yourname_Rollnumber_Assignmentnumber.zip

Task 1: Write a C Program to reverse strings received from 5 clients. [10.5 marks]

Build connection-oriented (TCP) client server model. Create 5 clients and every client sends its string to the server and the server reverses the string received from the clients and sends its corresponding reversed string back to every client.

Action at Server :

1. Create a socket using socket( ) system call. [0.5 mark]
2. Bind server's address and port using bind( ) system call. [0.5 mark]
3. Convert the socket into a listening socket using listen( ) system call. [0.5 mark]
4. Wait for client connection to complete using accept( ) system call. [0.5 mark]
5. Receive the client request using recv() system call which consist of the name of the command that is to be executed along with data parameters (if any) [0.5 mark]
6. The command is interpreted and executed. [0.5 mark]
7. On successful execution the result is passed back to the client by the server.

Actions at every Client [1.5*5=7.5]

1. Create a socket. [0.25 mark]
2. Fill in the internet socket address structure (with server information). [0.25 mark]

3. Connect to the server using the connect system call. [0.25 mark]

4. The client passes the command and data parameters (if any) to the server. [0.25 mark]

5. Read the result sent by the server, write it to standard output. [0.25 mark]

6. Close the socket connection. [0.25 mark]

Sample Inputs: i) The Client1 sends the string “IIT Ropar” to the server.
ii) The Client2 sends the string “CSE Dept” to the server.
iii) The Client3 sends the string “CN LAB4” to the server.
iv) The Client4 sends the string “Socket Program” to the server.
v) The Client5 sends the string “Implemented in C” to the server.

Sample Outputs: i) The Client1 received reverse string as “rapoR TII”
ii) The Client2 received reverse string as “tpeD ESC”
iii) The Client3 received reverse string as “4BAL NC”
iv) The Client4 received reverse string as “margorP tekcoS”
v) The Client5 received reverse string as “C ni detnemelpmI”

Task 2: Write C programs, one for TCP server (handles requests for multiple users) and other 5 programs for 5 clients to build a **railway ticket reservation system** using a server-client model. Your server program will be a multi-process server (e.g. server will use a “fork” process for every new client it receives or you may use some other methods). Multiple clients should be able to simultaneously chat with the server and the connection should be persistent. You may use multithreading using **pthread or fork()** system call to implement the server. [9.5 Marks]

The protocol between the client and server is as follows:
1. The client connects to the server, and then asks the user for inputs. The user enters their required type and quantity of the tickets. The user's input is sent to the server via the connected socket.
2. Assume that the server is an agent and the clients are the customers.
The agent starts the ticket booking by login with the valid user id and password. Then, the agent reads the user's input from the client socket, evaluates the availability of the tickets, and sends the result back to the customers as well as writes the following in a file named “server_records.txt”. [at the beginning create an empty file]. Make sure to handle the race conditions while writing to this file at the server.
3. Every customer should display the server's reply, and prompt the next input, until the customer’s program terminates.
4. The agent should be able to handle all required operations to implement the **railway ticket reservation system**.

NOTE: i) For task2 marks will be given based on your implementation.
ii) Don’t use curl commands for making the clients.