Introduction

Most inter-process communication uses the client server model.

The client process connects to the server process typically to make a request for information.

Sockets provide the communication mechanism between two computers using TCP/UDP.

Stream sockets use TCP which is a reliable, stream-oriented protocol, and datagram sockets use UDP, which is unreliable and message oriented.

Creating Socket on Server Side

Create a socket with the socket() system call.
Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
Listen for connections with the listen() system call.
Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
Send and receive data using read() and write() system calls.

Creating Socket on Client Side

Create a socket with the socket() system call.
Connect the socket to the address of the server using the connect() system call.
Send and receive data using read() and write() system calls.

Problem Description-

Problem 1-

Write two separate C program, one for TCP server and one for TCP client in which server listens on some port, client connects to server sending some arbitrary message and server acknowledges it.

Problem 2-

Write two separate C program, one for TCP server (handles request for single user) and other one for client.

At server side-

Creates a socket and listens on some specific port to process client request.
There is a default file present having n lines and the server should be able to process EVALUATEX and WRITEX request from the client.

1. The server process should tokenize string received from the client that may contain EVALUATEX or WRITEX request in following format-

   ○ **EVALUATEX k**- read k^{th} line from the starting of file, evaluate the expression and return result to client.

   Algorithm for Expression Evaluation-

   Create 2 stacks for arithmetic expression evaluation, one value stack to store the numbers and another operator stack to store the operators.

   1. While there are still tokens to be read in expression,
      1.1 Get the next token.
      1.2 If the token is:
         1.2.1 A number: push it onto the value stack.
         1.2.2 A left parenthesis: push it onto the operator stack.
         1.2.3 A right parenthesis:
            1 While the thing on top of the operator stack is not a left parenthesis,
               1 Pop the operator from the operator stack.
               2 Pop the value stack twice, getting two operands.
               3 Apply the operator to the operands, in the correct order.
               4 Push the result onto the value stack.
            2 Pop the left parenthesis from the operator stack, and discard it.
         1.2.4 An operator (call it thisOp):
            1 While the operator stack is not empty, and the top thing on the operator stack has the same or greater precedence as thisOp,
               1 Pop the operator from the operator stack.
               2 Pop the value stack twice, getting two operands.
               3 Apply the operator to the operands, in the correct order.
               4 Push the result onto the value stack.

2 Push thisOp onto the operator stack.

2. While the operator stack is not empty,
   1 Pop the operator from the operator stack.
   2 Pop the value stack twice, getting two operands.
   3 Apply the operator to the operands, in the correct order.
   4 Push the result onto the value stack.

○ **WRITEX expr**- append expression string to the end of file present at server and return “SUCCESS!!” to the client as acknowledgement.

expr (arithmetic expression) shall be appended only if it doesn't exist in file, else return appropriate message to the client.

At client side -

1. Client process should take input from the user whether to EVALUATE or WRITE on the server side.
2. It then initiates connection to server and forwards the query to server.
3. Receives output from server and displays it to the user.

Marking Scheme

Total - 100 Marks.

Problem 1-

Establishing connection between server and client - 15 Marks.

Problem 2-

For handling EVALUATEX queries - 40 Marks.
For handling WRITEX queries - 15 Marks.
Error handling strategies- 20 Marks

Coding style - 10 Marks.