Assignment - 2

Introduction

Most interprocess 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 (handles request for multiple users) 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 READX and WRITEX request from the client.

1. On receiving a client's request server should fork a separate process to handle specific client's request.
2. The server process should tokenize string received from the client that may contain READX or WRITEX request in following format-
    - **READX k**- read k^{th} line from the starting of file and return to client.
    - **WRITEX msg**- append msg string to the end of file present at server and return "SUCCESS!!" to the client as acknowledgement.

At client side-

1. Client process should take input from the user whether to READX or WRITEX 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 - 75 Marks.

Problem 1-

Handling Concurrency - 15 Marks
For handling READX and WRITEX queries - 25 Marks.
Error handling strategies- 25 Marks

Coding style - 10 Marks.

Reference Links

1- **Blocking vs Non-blocking call**
https://www.scottklement.com/rpg/socktut/nonblocking.html