Operating Systems Lab (CS 470):

Lab 3: Write a separate client and a server program in C/C++ using TCP/IP protocol. Simulate an airfare selling system, where the server is in charge to keep the administration of the tickets, while the clients can connect to the server and ask to purchase the available tickets.

Overview

Communication between two software programs can be done using sockets among others. A socket is identified by an IP address concatenated with a port number. The server waits for incoming clients requests by listening to a specified port. Once the request is received, the server accepts a connection from the client to complete the connection. Servers which implement specific services such as ftp, telnet, http listen on some dedicated ports (telnet (23), ftp (21), http (80), therefore use port numbers bigger than 1024 for that purpose. Ex. use 5432 – probably nobody is using that.

Instructions

1) Both the server and the clients program should be written in C/C++ under Linux/Unix.
2) The client program(s) should connect using socket to the running server program.
3) If the server is not available (not running, connection problems, etc.) the client should timeout (try several connection attempts in some given interval (see Timeout) in the ini file) and exit.
4) The connection IP address (server), the port number and the timeout should be read from a regular text file at each client [re]start.
5) For each client the server should listen in a separate thread (Not mandatory but highly recommended!)
6) Initially, when the server program starts all the tickets are available for sale. The tickets should be allocated from a rectangular map, where each pair of (column, row) represents one particular seat. The size of the map (# of rows, # of columns) should be provided as command line arguments when the server is launched. If there are no such parameters given, some default values (ex. 10x10) should be considered.
7) The client software can act:
    a. Each time when a new purchase is to happen we have to introduce from the terminal the row and column for the seat we want to purchase. If the seat is available and valid (see row and column), the server will allocate it and will notify the client or otherwise send the corresponding [error] message. A second purchase cannot be initiated from the same client until the first one is not closed successfully or unsuccessfully. This process can go on and on until all the tickets are sold. In case of asking for an erroneous seat, the client should be notified.

b. The client can ask for random seats, generating the row and columns randomly. This process is repeated –with some delay (see sleep(10)) until all the seats are allocated in the plain.
c. The operation a) and b) should be launched from command line when the client is starting. If in the command line there is “manual” the manual way of purchasing is activated, if “automatic” is invoked the client should purchase tickets automatically.

Notes

- When the client connects first to the server, the server is sending the client the initial size of the map (see # of row, # of columns).
- Each operation should be traceable on the clients’ side and the server’s side. We should be able to identify which client is sending what, and what the server is answering each time to the client(s) purchase requests. We also see the current status of the seat map.
- The *ini* file (containing the *ip*, *port*, n and *timeout*) used by the client should be given as command line parameter. If no file is some default values should be considered. The configuration of the ini file should be:

  IP = x.x.x.x

  Port = z

  Timeout = y

Rubric

Task	**Points**
Error handling	2
1 Client/ 1 Server	4/4