Operating Systems Lab (CS 470):

Lab 4: Write in Linux 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 (see more details below) and exit.
4) The connection IP address of the server, the port number and the timeout should be read from a regular text file at each client [re]start (see details below).
5) For each client the server should listen in a separate thread (Not mandatory but highly recommended!) The software will be tested at least with two clients connected simultaneously to the server application.
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. Ex.: ***./server 10 15*** means that the server program will distribute tickets from 10 rows, each row containing 15 seats.
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 without disconnecting the client 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 column pairs randomly. This process is repeated –with some delay random delay of 3/5/7 seconds) until all the seats are allocated.
c. The operation a) and b) should be launched from command line when the client software is started. 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. Ex. ***./client manual*** means that the client will try to connect using the default IP and Port number and the client will purchase the tickets by introducing at the standard input each time the row and column for the ticket to be purchased. If we call the client like this***: ./client automatic myinifile.txt*** in that case the client software will connect to the server application using the information coming from the ***myinifile.txt*** and the generation of the rows and columns for the tickets will happen fully automatically. In both cases the simulation runs or can run until all the tickets are sold. If that is the case the clients are disconnected from the server.

Notes

- The server side should print all the time a map indicating the current status of the seats.
- The *ini* file (containing the *ip*, *port* and *timeout*) should be given as command line parameter. If no file is given, use some default values. The ini file should follow the ini file structure used by Windows involving section names, names and values. A possible configuration of such an ini file could be:

  [Connection]

  IP = x.x.x.x

  Port = z

  Timeout = y

Rubric

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