Operating Systems Lab (CS 470):

Lab 4: Write a separate client and a server program in C/C++ using TCP/IP protocol. The server program generates randomly a rather huge number (close to INT_MAX, see for details *limits.h*), and the client(s) should implement the prime factorization. The prime factorization mechanism should be as follows:

1) The server is generating a base and a number (in that base) which is sent along with a prime factor (decimal).
2) The client should check if the number is divisible by the prime factor (in decimal) and send back the results to the server.
3) The server is distributing this prime factor checking jobs equally among the connected clients using similar mechanism as Round Robin for CPU scheduling.
4) The process ends when a) there are not clients connected, b) there were ***n*** consecutive prime numbers generated by the server. Each time the server is printing the prime factorization. Ex. 79860(10) = 2^**2***3^**1***5^**1***11^**3** where the 2, 1, 1, 3 is coming from the clients. Another example 101(2) = 5^**1**

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.
2) The client program(s) should connect via socket to the running server program.
3) If the server is not available (not running) the client should timeout (try several attempts at some given interval) and exit.
4) The connection IP address (server), the port number, the n parameter 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.
6) The base of the generated number should be a value between 2 and 10. As you generate the random number in decimal, you have to check if the number is correct in that base or not. If not correct generate a new number.
7) In order to follow the computation process each client is writing on the screen what value is receiving and what answer is sending back, and once all the prime components are checked the server is going to print the prime factorization results.

Notes

- The *ini* file (containing the *ip*, *port*, n 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
  [Other]
  n = v

Rubric

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