CENG 493 Special Topics in Computer Engineering: Cluster Computing

Homework 1/032

Regulations:

- Due Date: 30 March, 2004
- Submission: Electronically. You will be submitting your program source codes, Makefile and all necessary data files through a compressed tar file which you will name as **hw1.tgz** by using the homework submission link https://submit.ceng.metu.edu.tr
- Team: There is no teaming up. The homework has to be done and turned in individually.

Problem:

You are expected to write a simple client/server application that reads matrices from the clients and returns their multiplication. You will use udp/ip network sockets for the communication. The server will use threads for handling each connecting client. Programs should be written in C.

Server:
The server program, named MATMUL will be a server that listens to a given port. When a client connects to this port, it should create a thread for the new client and continue to listen for new connections. The new thread should open a new port and interact with the client using that port.

The multiply program can take three arguments:
1. A port number. If no port number is given, the default value should be used which is **1234**.
2. A log-file name. If no log file name is given, the default value should be used which is **MATMUL.log**
3. The number of maximum allowed simultaneous connections. If no maximum connection number is given, the default value should be used which is **5**.

Server is supposed to hold a single log file. The content of this file should include the following:
1. Date of request
2. IP of the client
3. Request name (One of the 7 possible commands a user can give to

the client program.)

For each request from the clients, a line should be written to this log file. The order should be identical with the time the requests are received.

Client:
The client program, named **client** will be a client that connects to a given port on a given host. Once connected, the server will assign a new port for communication. From the new port, the student program can issue some commands and receive answers.

The **client** program can take two arguments:
1. A port number. If no port number is given, the default value should be used which is **1234**.
2. A hostname. If no hostname is given, the default value should be used which is **localhost**.

Client program should be interacting with the user. It should read the inputs of user from stdin and write the output to stdout. There are 7 different commands a user can give:

DIM1 *num* /* Number of rows of the first matrix*/
DIM2 *num* /* Number of rows of the second matrix*/
ROW1 *n num num ...* /* Contents of the **n**th row of first matrix.*/
ROW2 n num num ... /* Contents of the **n**th row of second matrix.*/
MULTIPLY /* Tell the server to multiply the first and second matrices */
RECEIVE /* Receive the multiplication of the first and second matrices. */
QUIT /* Close connection and quit client */

When received one of these commands from the stdin, the program should execute the appropriate action and report the outcome to user from stdout. All possible outcomes are:
- **DIM1** *num* command can return
  - **OK** if *num* is a positive integer and smaller than or equal to 5 and received acknowledge from server
  - **ERROR** is *num* is not a positive integer or is bigger than 5
  - **CONN** if the server response timed out
- **DIM2** *num* command can return
  - **OK** if *num* is a positive integer and smaller than or equal to 5 and received acknowledge from server
  - **ERROR** is *num* is not a positive integer or is bigger than 5
  - **CONN** if the server response timed out
- **ROW1** *n num num ..* command can return

    - **OK** if *n* is a positive integer smaller than 6 and there are **DIM2** number of integers after *n and received acknowledge from server*
    - **ERROR1** if *n* is not a positive integer or is bigger than 5
    - **ERROR2** if there are not exactly **DIM2** number of integers after *n*
    - **CONN** if the server response timed out
- **ROW2** *n num num ..* command can return
    - **OK** if *n* is a positive integer smaller than 6 and there are **DIM1** number of integers after *n and received acknowledge from server*
    - **ERROR1** if *n* is not a positive integer or is bigger than 5
    - **ERROR2** if there are not exactly **DIM1** number of integers after *n*
    - **CONN** if the server response timed out
- **MULTIPLY** command can return
    - **OK** if the the given matrices are multiplied by the server
    - **ERROR** if there are missing data (contents of a row not given)
    - **CONN** if the server response timed out
- **RECEIVE** command can return
    - The multiplication of the first and second matrices as a **DIM2**x**DIM2** matrix
    - **CONN** if the server response timed out
- **QUIT** command can return only "CONNECTION CLOSED" after closing the connection
    - **CONN** if the server response timed out

The commands and their answers for the client program are strict. But you are free to chose the nature of communication between the server and client as you like. The internal design of packages these programs will use is up to you.

Notes:

- The dimensions of the first matrix is DIM1xDIM2, while the dimensions of the second matrix is DIM2xDIM1.
- Because of UDP/IP, packages may drop on the way without notice. So you should implement some sort of timeout mechanism while waiting for an answer from the server. If timeout is exceeded, notify the user. The timeout period for this homework should be 60 seconds.
- In this homework you should assume that there will not be any corruption during transfer.
- Multi-Threaded program is like a multi-process program, but there are some important differences. Each thread have access to the global variables defined by the initial program. Also the static

variables can be accesses by all threads. So do not use global or static variables when not necessary. A change to a global variable by a thread will effect all other threads.

- You can use the "pthread" library which is available on Linux and sun machines.
- Provide your programs with a makefile that can compile them by a simple make command.
- Put all the necessary files in the tar file. Do not forget to compress the tar file with gzip command before sending.
- Direct all questions about the homework to the newsgroup of the course so all students can follow.