# Final Project Report

for

# System Programming CSE344

Spring 2018

Deniz Can Erdem Yılmaz – 151044001

Gebze Technical University

# 1. Problem

There are two main problems that should be solved for this project. Just like our homework #5 there is a seller person which serves customers for their requests. But the difference is customers are now have their own client program to send request to the seller which is server program. Since there are two programs there is also a need of some type of a communication line between them to let server and client understand each other. There are also some subproblems that eventually help or profit by this communication line. All these sub-problems will also be covered under the Problem Solution Approaches section.

# 2. Problem Solution Approaches

In this section all of the problems will be listed under their associative program and function name for the clarity. The functions or problems that are mutual for both programs will be discussed under a third section to prevent repetition.

## 2.1 Server

Server is the program that waits for connections from client programs and redirects the requests to the threads that act as providers whose information will be read from a data file. Program needs 3 arguments while starting the program: port number for connection, a file name to read provider informations from and another file name to log the statistics of the providers at the end of program. In any case of error server will print a proper informative message to the screen and will terminate itself. This may affect the clients which are connected to and/or waiting response from server during the failure and clients that try to connect to server during or after the failure. Even if all the providers log off the server will be up and running until a termination signal (SIGINT) arrives from the end user.

### 2.1.1 ParseFile

ParseFile function is the first function that starts processing at the beginning of the server. Purpose of the function is reading data file to create provider threads according to provider informations data file stores. It has 3 parameters which are a name for file to read, an array (type of struct Provider which holds various informations about providers) to hold the data read from file and an integer pointer to inform the caller about the size of array. Function’s return value indicates whether the task was successful or not.

Since the structure of the provider data file is given, struct Provider defined to hold all the information in one place. To read the file easily getline() and sscanf() functions used. First of all the storage array that provided as parameter initialized as an empty array of this struct type. Then, in a loop, every line read from file and fields of struct filled using sscanf() from the line. At the end function returns 0 to indicate everything is fine.

### 2.1.2 SearchForProvider

SerachForProvider function is a helper function for the RedirectorFunction. Purpose of the function is finding the provider fits best to the request of client to redirect request. It takes a struct Request type parameter to analyze the request to find the perfect provider. It works just like a basic linear search. Sets the result to lowest index and iterates through each provider to see if the next one is better than the current best.

To determine the best provider 3 comparisons made: first the provider should have lesser than 2 clients waiting in queue, the provider should be online and the providers value of prioritized stat should be better than current best. There are 3 stats that can be prioritized: quality that compares performance of provider, speed that compares the queue size of provider and cost that compares the price of provider. Index of the best fit provider returned as the result. If there is no online providers or all of their queues are full error code will be returned to inform client that their request can’t be processed right now.

### 2.1.3 DoHomework

DoHomework function is a helper function for the ProviderFunction. Purpose of the function is finding the result of homework requested by a client. It takes the degree value of homework as parameter. It uses my homework from CSE102 course to calculate the cosine value of given degree using Taylor series with approximation step number of 15. Functions factorial, power and cosine will not be defined in this report since they already exist in standard math libraries.

### 2.1.4 ProviderFunction

ProviderFunction is a function used to start a thread with. This function simulates the work of a provider and is one of the main functions of server. It takes an index as parameter to get the informations of the provider from the struct Provider array that defined as global array and filled by the ParseFile function.

Function starts by getting locks for online provider array to declare itself as online. To declare itself online it increases the onlineCount variable by 1 and sets the online field of the Provider struct on it’s index as TRUE. This is done so SearchForProvider function can check if the provider is open to new clients.

After that an infinite like loop starts until either server shuts down or the time is up for the provider. This loop basically waits until a client assigned to the provider. This waiting is done by waiting on condition variable. Since providers will be stay online for a specific time, pthread_cond_timedwait() used to see if the time is up or not. This loop ensures the created threads will stay there to assign jobs after jobs (which is the basic concept of thread pools) and also helps to find out when the time is up.

When a thread will wake up from waiting the condition variable it will check the return value of pthread_cond_timedwait(). If return value is error code and errno is set to ETIMEOUT it means time is up for that provider and it should get away from the infinite loop. If error code is not ETIMEOUT then there is an unidentified problem so it will print error message and quit the program. If return value indicates there is nothing wrong then it means there is a client assigned.

When there is a client assigned to a provider and it is awake, provider will get the client from it's queue and then release the mutex lock. The queue holds informations about client's socket and request. Using DoHomework function the result of the homework will be calculated to get it back to the client. To simulate doing the homework thread will sleep for a random time between 5-15 seconds. After sleeping is done provider will send a response to the client using the socket came from queue. When the send operation is done the statistics are recorded to the global stat array that holds statistics for each provider to use while logging. Finally the client socket will be closed and the loop will be rewind to beginning.

If time is up or the server terminates, which causes the while loop to break, provider will get the mutex lock for the online status variables just like the beginning to declare itself as offline. To prevent clients wait infinitely, before returning the provider empties the client queue and sends each of them a response with a error code that specify why the connection is ending and their homework is not done. It will also print a message to the server screen informing provider name and the reason of quit and then quits the function.

### 2.1.5 RedirectorFunction

RedirectorFunction is a function used to start a thread with. This function is also one of the main functions of server and is like a bridge between main thread which is the main() function and the ProviderFunction thread. It takes a socket as parameter to receive the request from a client that accepted by the main thread.

First of all it reads all information the client wrote to the socket to fill the request struct. Then it gets all the mutex locks for provider queues to make a search between providers using SearchForProvider function to find the perfect provider that request looks for. If SearchForProvider returns an error code then RedirectorFunction sends a response that holds an error indicator and error code to notify the client request is failed to complete.

If the SearchForProvider function was successful redirector will create a work struct that holds the request informations and client socket. Assigns client to the found provider by inserting it to the work queue provider and signals the provider using pthread_cond_signal(). Then releases the locks and exits function.

### 2.1.6 LogStats

LogStats is a helper function to log the current statistics of the providers to the log file. It takes an integer value to inform if logging is requested due to an error or end of program. It prints the name, working time and total number of clients for each provider to the log file whose name was given as command line argument to the program itself. If status indicates the error then it will be indicated at the end of log file too.

### 2.1.7 SignalHandler

SignalHandler is a function to assign as the handler of SIGINT function. Whenever SIGINT signal arrives to program it will change the flag that indicates continuation of the server to the FALSE. So the loops of provider threads and the main thread will come to a stop to terminate the program.

### 2.1.8 Main

Main function of the server is uses all the functions above to build the server structure that acts as a multi threaded, with threadpool, server waits for connections from clients to do calculations and returns the answer back to them. It does basic initializations of global variables, mutexes and condition variables at the beginning and destroys, deallocates them at the end of program to clean up.

Besides the preparations, it also has its own infinite working loop to accept clients from socket and call RedirectorFunction to redirect the incoming client to a suitable provider. Once a client connected it will first check if there is any providers left online or the thread count exceeded. There shouldn't be a thread limit to server. But since I somehow failed to dynamically resize the array that holds thread IDs I used a static sized array. If all providers are offline or the thread limit exceeded the main function sends a response directly to the client that indicates the request has failed without creating a redirector thread.

At the end of the main, after joining all threads LogStats function called with safety code to print the statistics to the both log file and to the server program screen to inform the user about provider statistics.

## 2.2 Client

Client is the second program for the to communicate with server to request the answer of homeworks to act as students. The work load is much lesser than the server itself since it just sends homework informations, waits for response and prints back the result to the screen. Client doesn't have any client specific functions aside from the main itself. So the remaining functions will be covered at the Common section.

### 2.2.1 Main

Main function of client creates a connection point between itself and the server whose address informations are given as command line arguments. Then prepares a request struct to send over socket to the server and then sends it. After sending the request it immediately waits for receiving the response from server. Once the response has received it prints the informations about provider, time spent and result to screen to inform the user and quits.

### 2.3 Common Functions

These are the common functions that are used by both server and client

#### 2.3.1 CreateConnection

This function uses POSIX socket.h library to create a socket. There are slight differences for server and client versions since one is creating socket to listen incoming connections and the other is creating socket to connect other sockets. For the server it uses socket() - bind() - listen() order while client uses socket() - connect() order to let them communicate with each other. Once the sockets are created server returns the socket descriptor to let the caller accept connections or read/write data from/to socket.

#### 2.3.2 ReceiveAll

RecieveAll function is a helper function to receive a struct from the socket. It has 3 parameters: a socket to receive from, a struct pointer to use as buffer and the size of buffer. It is basically similar to/same as read() and recv() system calls. The reason of wrapping is to make sure all the bytes are required are read from socket. It reads in a loop until a total of *size* bytes read from socket to buffer given as parameter.

#### 2.3.3 SendAll

SendAll is another helper function just like ReceiveAll, a wrapper to the send() system call to make sure all *size* bytes are send to the server.

## 3. References


- https://stackoverflow.com/questions/4246588/fprintf-like-function-for-file-descriptors-i-e-int-fd-instead-of-file-fp
- https://stackoverflow.com/questions/8304259/formatting-struct-timespec
- https://stackoverflow.com/questions/13479760/c-socket-recv-and-send-all-data?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
- https://stackoverflow.com/questions/1486833/pthread-cond-timedwait
- https://stackoverflow.com/questions/2150291/how-do-i-measure-a-time-interval-in-c
- https://github.com/Layso/CSE102-Homeworks/blob/master/HW04/HW04_151044001_Deniz_Can_Erdem_Y%C4%B1lmaz_part1.c
- The Linux Programming Interface – Michael Kerrisk
- CSE344 lecture notes