Gebze Technical University
CSE344 Systems Programming Course
-
FINAL REPORT

Muhammed OZKAN 151044084

June 9, 2021

Abstract

You are expected to write 2 programs: a client and a server. The server will run as a single instance,will open up a stream socket and wait for clients to connect and conduct SQL queries. The client can/will run as multiple instances (i.e. there will be more than one client processes running concurrently), read its SQL queries from a file, send them to the server, and print the received responses. Once all their queries are done, the clients will exit.

1 Overview

The assignment I submitted is a complete assignment. It supports all the requirements and parameters requested in the homework document and server client programs run according to the given parameters. The server reads a csv file and writes its output to a log file. Server is a daemon and cannot run more than once. The server also has a thread pool with the number specified in the parameter. It communicates with socket. Implements the reader writer paradigm and prioritizes writers. It has a dynamic storage structure. The client, on the other hand, reads its own queries from a file and in turn sends them to the server via socket. Multiple client program can work at the same time.

2 How did I solve this problem.

First of all, our programs check the parameters given to them, if there is no error, the program continues to run. Then the server uses a named semaphore so that it is not run more than once. After that, the server turns itself into a daemon and writes the outputs to the log file after this stage. Variables to be used in the thread pool are created. socket structures to be used for communication are created. Threads are created. Mutex and condition variables are used in the synchronization of threads. Finally, the CSV file is read and placed in memory. After this stage, a query is expected from the clients via socket. Each incoming query is received by the main thread and sent directly to one of the threads in the pool. According to the reader writer paradigm according to the query type, it reads or writes the data in the memory and sends it to the client. Each client sends their queries sequentially until their queries are finished and prints the incoming answers on their screens. Clients whose processes are finished are terminated. The server still continues to run.

3 My design decisions.

The program I prepared uses a dynamic data structure. It frees up memory for each character it reads and places it there. Each of the threads is waiting for their job to be given behind a condition variable of their own. The system, which works according to the readers writers paradigm, gives priority to the authors. Data communication is done through sockets. Clients read their own queries from the file and send them to the server according to a certain format and protocol, and read until the replies are finished. The program is case sensitive in SQL queries. SQL keywords must be uppercase. It supports all SQL commands specified in the assignment file.

```
typedef struct CSV_FIELD
{
char *text;
size_t length;
} CSV_FIELD;

typedef struct CSV_BUFFER
{
CSV_FIELD ***field;
size_t rows;
size_t *width;
char field_delim;
char text_delim;
} CSV_BUFFER;
```

4 What requirements did I meet and which did I fail?

All requirements are fully met. Due to the data structure and algorithms used, the program takes longer to respond to queries as the data size grows. All dynamic variables created at the end of the program are deleted and resources are returned to the system. Open source codes are used for CSV parsing in the program, the link is available below. Below is the source used to make a daemon. The following links were used for the time stamps and time measurement used in the printouts.

5 Source Sites Used

https://nullraum.net/how-to-create-a-daemon-in-c/
https://github.com/winobes/libcsv
https://stackoverflow.com/questions/1444428/time-stamp-in-the-c-programming-language
https://stackoverflow.com/questions/656542/trim-a-string-in-c
https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/